ETH zürich

PAUL SCHERRER INSTITUT
PSI

Model-Agnostic Machine Learning Explainability for Regression Tasks in Physical Systems

Master Thesis
Department of Physics
ETH Zürich

Written By
F. Wilke Grosche

Supervised By
Dr. A. Adelmann (ETH, PSI)
Prof. Dr. Ce Zhang (ETH)

Collaborators
Arnau Albà
Cédric Renggli

11/10/2021

Abstract

Black-box machine learning is a tool that has widespread applicability. In this work, we examine a method used to evaluate these models and their decision making processes. We apply the explainability techniques SHAP [1] and LIME [2] to models of physical systems.

We model the damped and driven Duffing Oscillator [3] as a toy model on which to test the explainability methods. The modelling is performed with numerical integration as well as two machine learning models inspired by the network proposed by Renato Bellotti in [4].

We then apply the explainability techniques to these models and compare them to XGBoost [5] and the gradient as a measure for feature importance. The feature importance is a metric designed to tell us how much a given feature contributes to the output of a black-box model.

Additionally, we test the explainability methods SHAP and LIME on a version of the toy model for which features have been added to the training data. An irrelevant feature is identified reliably by SHAP and to a lesser extent by LIME. Highly correlated features impact the explainability of the machine learning models but not the simulation.

We conclude that SHAP succeeds in extracting realistic feature importance for these systems. It can be used to gain information about relationships between model input and output. Therefore it can be used to understand the model's predictions and for the purposes of feature reduction and model optimisation. LIME is inferior to SHAP in this aspect for this application to the Duffing Oscillator in all test cases, due to to its dependence on being properly tuned, without which it does not deliver reasonable feature importance. XGBoost delivers aggregate feature importance that aligns with that of SHAP.

All generating code can be found under `https://github.com/wgrosche/XAI`.

Acknowledgements

Special thanks goes to Arnau Albà and Cédric Renggli for their consistent input and helpful advice throughout the research phase and the process of writing this thesis. Thanks also to my family for their unwavering support throughout my studies.

Contents

1 Introduction **4**

2 Theory **5**
2.1 Explainability . . . 5
2.1.1 Local Model-Agnostic Explainability . . . 5
2.1.2 LIME . . . 5
2.1.3 Shapley Values . . . 7
2.1.4 SHAP . . . 8
2.1.5 Numerical Explainer . . . 9
2.2 Duffing Oscillator . . . 9
2.2.1 Numerical Integration . . . 10
2.2.2 Data Generation . . . 10
2.2.3 Simulation . . . 11
2.2.4 Duffing Oscillator Parameter Configurations . . . 11
2.3 Neural Networks . . . 14
2.3.1 Data Preprocessing . . . 14
2.3.2 Deep Neural Network . . . 14
2.3.3 Shallow Neural Network . . . 15

3 Methods **17**
3.1 Hyperparameter Tuning . . . 17
3.1.1 LIME . . . 17
3.1.2 Gradient Step Size . . . 19
3.1.3 Neural Networks . . . 20
3.2 Network Training . . . 21
3.2.1 Lightly Damped Double Well . . . 21
3.2.2 Heavily Damped Double Well . . . 23
3.2.3 Single Well . . . 24

4 Results **26**
4.1 Using only (x_0, v_0, t) as Training Features . . . 26
4.1.1 Lightly Damped Double-Well . . . 26
4.1.2 Heavily Damped Double-Well . . . 29
4.1.3 Single Well . . . 37
4.1.4 Section Summary . . . 38
4.2 Adding an Irrelevant Training Feature . . . 38
4.2.1 XGBoost . . . 41
4.2.2 Section Summary . . . 43
4.3 $g(x, v)$ as a Training Feature . . . 43
4.3.1 Section Summary . . . 46
4.4 Adding Global Model Parameter γ as a Training Feature . . . 46
4.4.1 Networks . . . 46
4.4.2 Explainability . . . 47
4.4.3 Section Summary . . . 51

5 Discussion and Further Work **52**

6 Conclusion **54**

A Appendix A **57**

Chapter 1

Introduction

Machine learning models are becoming an increasingly important tool in decision-making processes due to their simplicity and adaptability. Whilst they require a large amount of data and computation in order to train, they can be vastly more efficient in making predictions than conventional methods. As discussed in the Master thesis by Renato Belotti [4], conventional methods used to make predictions in physics often require numerical integration and, as such, many iterations before predictions can be made about times in the future. With machine learning models this is not an issue as the prediction is turned into a series of linear transformations. These linear transformations are computationally cheap.

Such a tool can find uses in experimental physics. The goal could be to reduce the space of experiments required to investigate a phenomenon. Even if the accuracy of the tool is poor it may aid in reducing the number of configurations to test by orders of magnitude. Promising configurations can then be simulated as before. An example of such an application is a deep neural network designed to classify superconducting compounds [6].

One problem with machine learning is that the methods most successful at some complex tasks, such as the deep neural network in the aforementioned example, are what is known as black-box models [7]. This means that, when faced with a prediction made by this model, we do not know the decision-making process. Knowing this process has many benefits. We could use it to infer the veracity of the prediction. We could use it to gain further knowledge of the data we are modelling and the underlying system. And we could use it to adapt our model, to more efficiently use the data.

Explainable machine learning is a method to understand the decision-making process of a black-box machine learning model. What we get from explainable machine learning is an idea of which input features contribute most to a prediction. We call this the feature importance. From this we can infer the decision-making process, giving us access to the benefits of white-box models without having to sacrifice the ability to deal with the complex problems that black-box models excel at. Explainable machine learning has found applications in the medical field such as in [8].

In this work, we will be examining the application of explainers for a regression task that exhibits oscillatory behaviour, the Duffing Oscillator. The questions we will be looking to answer are:

- Are explainers able to replicate a human's understanding of an oscillating system?
- Are explainers able to discern a pattern from a system that is not intuitively explainable? Can we gain knowledge of a system's workings that we can't get by just looking at the data?
- Can we use explainers to evaluate a model's accuracy? Can we judge whether a model is likely to perform well based on the explainability?
- Can explainers be used to refine the feature space? Can they be used to remove features without significantly impacting model performance?
- How do the established explainability frameworks compare in their performance?

To answer these questions, we will be designing a neural network to model the Duffing Oscillator [3] and predict the trajectory of a point up to a time t. We will then use the explainability methods SHAP and LIME to analyse the models and extract information about feature importance. SHAP and LIME function by building an interpretable surrogate model to the one used to model the system. This model can then be evaluated to gain an idea of the feature importance. We will conduct this process with different input features, such as the initial system energy, which we choose to test whether the explainers can determine their involvement in the prediction process.

The goal of this work is to see whether explainable machine learning has promise in its applications to physical systems for the purposes of supplementing human understanding of a system and facilitating the refinement of the machine learning models used to model physical systems.

Chapter 2

Theory

2.1 Explainability

Explainability is an attempted solution to the black-box problem [7]. It differs from the alternative "Interpretable machine learning" in that we keep the black-box model, but try to probe it in order to infer the decision-making process. Interpretable machine learning works by using a model from which the decision-making process can be inferred directly, examples of this kind of method are decision trees or linear regression models. Both of these methods aim to increase the trust in the predictions the model makes by understanding the process behind the decisions. Another method to do this is probabilistic machine learning. Models such as Bayesian Neural Networks [9] associate each prediction with the confidence the model has in the prediction, this serves a similar role.

First we will discuss why the black-box problem exists. Models such as deep neural networks use a huge number of trainable parameters in order to model the data. Image classifiers such as the Xception [10] network have of order 10^7 trainable parameters. This makes it impossible for even an expert in the field to extract the decision-making process of the network. Models that are easier to interpret do exist, the traditional linear regression model, or even decision trees, can be looked at and the process can be inferred. For example, we can look at the linear model

$$\hat{f}(x) = \beta_0 + \beta_1 x_1 + ...\beta_p x_p, \tag{2.1}$$

at a point x, where β_j is the weight for feature j, and infer a relationship between input and output. We do this by assigning a contribution

$$\phi_j(\hat{f}) = \beta_j x_j - \beta_j E(X_j), \tag{2.2}$$

to the j-th feature. Here $E(X_j)$ is the mean of the value for x_j.

Explainable machine learning is an attempt to get this functionality out of the black-box model. Instead of trying to explain the whole model, we attempt to approximate it locally with an interpretable model. This could be a linear regression model. We call this local approximation the "surrogate model". This surrogate model can then be interpreted in order to make statements about the black-box model locally. The methods used for generating the surrogate model differ from explainer to explainer. We will be looking at local model-agnostic explainers.

2.1.1 Local Model-Agnostic Explainability

Model-agnostic explainability methods do not care about the structure of the underlying model. They approximate the model around a point x by passing perturbations on that point through the model. They subsequently use these data to train their local surrogate model. We will be taking a deeper look at Local Model-Agnostic Models such as SHAP and LIME. These give us the "feature importance" for a single instance. The "feature importance" is a measure of how much a feature contributes to the prediction of a label. There are aggregation methods to group these values to get global feature importance across the whole dataset.

2.1.2 LIME

Local interpretable model-agnostic explanations (LIME [2]) functions by forming an interpretable surrogate model at the data point (x) being explained. This model is trained by sending perturbations on the data point through the model and looking at the output. In order to decide on which points should be used to train LIME, we define a neighbourhood (π_x) of the instance x. This is usually defined so that points that are further from

x have lower weights. This means that they are less important when training the surrogate model at point x. The formulation of LIME boils down to the minimisation problem:

$$\Phi(x) = \underset{g \in G}{argmin}\ L(f, g, \pi_x) + \Omega(g). \tag{2.3}$$

Where Φ is the explanation attributed to x. Here we minimise the error that the local surrogate model g has with respect to f in an area around x defined by the neighbourhood π_x. G is simply the family of all possible explanations. π_x is the neighbourhood of point x we use in training the surrogate model.

In order to make the model easier to explain we also minimise the model complexity $\Omega(g)$ in order to keep the number of features low.

This minimisation problem is tackled in the following fashion:

1. Choose a point of interest x
2. Draw values from the neighbourhood π_x
3. Put these values through the black-box model f
4. Weight samples by their proximity to x
5. Train a ridge regression model g(x) on these values
6. Interpret this model to explain the prediction f(x)

The weighting of points used in training the model is done by using a kernel which we discuss in Sec. 3.1.1. We will be applying LIME to Tabular Data. Here π_x is determined by sampling from a normal distribution which is subsequently scaled by the mean and variance of the training data found in the table. This is shown in Figure 2.1.

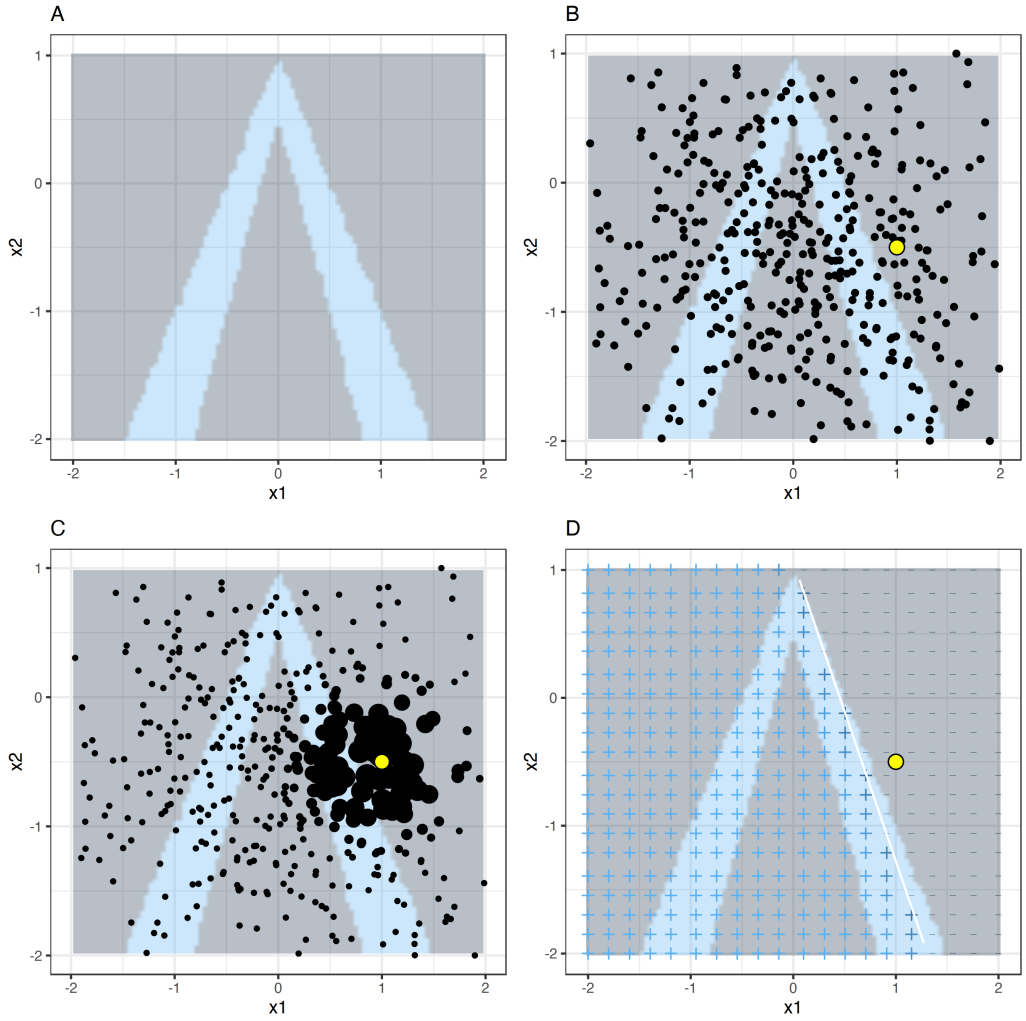


Fig. 2.1: A visual explanation of the selection and weighting of data points used in training a LIME explainer at point x (yellow circle). Top Left: The classification task with two classes, positive is light blue, negative is grey. Top Right: Points sampled uniformly at random from all available data. Bottom Left: Points weighted based on their proximity to x. Bottom Right: Predictions of the local surrogate model (white line). [11]

This figure highlights that an important aspect of training a LIME explainer is tuning it to fit the problem. If the neighbourhood had been larger then the decision boundary may have been placed elsewhere. With a smaller neighbourhood the classifier may have decided that everywhere is classified negative.

The concept of LIME is quite simple. We train a local approximation of the model with data passed through the model itself. The implementation ([12]) requires defining what constitutes a "neighbourhood" of x, and also what we should use as a surrogate model. For the decisions made in this work, refer to Sec. 3.1.1.

2.1.3 Shapley Values

Shapley values [13] are another approach to assigning feature importance. When attempting to explain a black-box machine learning model there are some differences when compared to a simple linear model such as that in Eq. 2.1. With the black-box model, we don't have similar weights that can be used to attribute feature importance. Instead, we draw from coalitional game theory, where we have the concept of Shapley Values that are used in calculating a party's contribution to the outcome of a group effort. It is the contribution to the payout, weighted and summed over all feature value combinations:

$$\phi_j(x) = \sum_{S \subseteq \{1,...p\} \setminus \{j\}} \frac{|S|!(p - |S| - 1)!}{p!} (val(S \cup \{j\}) - val(S)). \tag{2.4}$$

S is a subset of all features, x is the instance we're trying to explain and

$$val_x(S) = \int \hat{f}(x_1, ..., x_p) d\mathbb{P}_{j \notin S} - E_X(\hat{f}(X)) \tag{2.5}$$

is the prediction for feature values in S that are marginalised over features not in S. This method is the only one that satisfies the properties of "Efficiency", "Symmetry", "Dummy" and "Additivity" [11].

This approach doesn't work in a plug and play fashion in machine learning for the obvious reason that simply removing a feature isn't a viable approach. This is because the machine learning model has been trained on all the features. In order to calculate the exact Shapley values, we would need to average over every possible feature combination. This is computationally unfeasible for large feature spaces as the computational cost grows exponentially in the number of features. We bypass this by resorting to a Monte Carlo approach. In order to estimate the Shapley value, we

- Select an instance to explain 'x'
- Select a feature j that we want to get the feature importance of
- Choose a number of times M to sample the background dataset X
- For all m = 1,...,M
- 1. Draw a random instance z from the set of all instances X
  2. Randomly permute features, placing $x_{(j)}$ at a random position
  3. Create new instance $x_{+j} = (x_{(1)}, ..., x_{(j-1)}, x_{(j)}, z_{(j+1)}, ..., z_{(p)})$
  4. Create new instance $x_{-j} = (x_{(1)}, ..., x_{(j-1)}, z_{(j)}, z_{(j+1)}, ..., z_{(p)})$
  5. Bring features back into the original order
  6. Compute $\phi_j^m = \hat{f}(x_{+j}) - \hat{f}(x_{-j})$
- Shapley Value $\phi_j(x) = \frac{1}{M} \sum_{m=1}^{M} \phi_j^m$

So we compare values that are an amalgam of a random data point and the instance that we are trying to explain. Determining $\phi_j(x)$, the contribution of feature j to $\hat{f}(x)$, requires exchanging a random number of features minus the feature x_j and comparing it to the same vector with x_j also exchanged. Averaging over many such values gives us $\phi_j(x)$. The larger we make M the closer we get to the exact value of the Shapley value. The variance of the Shapley value is dependent on the number of times we sample, thus, if we want good results, we need to sample many times. The benefit of this sampling method is that it performs better with a large background dataset than kernel SHAP [14] (see Sec. 2.1.4). This is the method we will be calling the sampling explainer and it is proposed in the work by Erik Strumbelj [15].

2.1.4 SHAP

SHapley Additive exPlanations are a combination of Shapley values and LIME, aimed to tackle the computational cost of the Shapley values. We use the LIME methodology of training a network with data points that we pulled from our model but instead of weighting these points with a kernel that falls off with increasing distance we instead weight them with the SHAP kernel:

$$\ker_x^{SHAP} = \frac{(M-1)}{\binom{M}{|z'|}|z'|(M-|z'|)}. \tag{2.6}$$

Here z' is a transformed feature vector that simply contains a 1 when a feature is present and a 0 when a feature is missing. M is the maximum coalition size and $\|z'\|$ the number of features in z'. This means that we weight points where one feature is missing or all but one feature are missing highest. Points, where an equal number of features are missing or kept, are weighted lowest. As before, we have no way of determining the model output for a missing feature. The same approach as for Shapley values is taken in order to determine the value of the "missing" feature. We simply sample from the background dataset and replace values as before (2.1.3). Instead of sampling randomly we first sample points with the greatest weight, assigned by the SHAP kernel ($\ker_x^{SHAP}$). We then weight samples with the SHAP kernel before applying LIME to get an interpretable linear model. This method is the kernel explainer explained in [1]. The implementation of these methods is documented in [14]. We will judge how good this method is at replicating the more brute force approach of sampled Shapley values.

2.1.5 Numerical Explainer

Being in the advantageous position of having access to the ground truth model, we can try to quantifiably assign a true feature importance to the model. The initial approach here was to simply perform a finite differences differentiation of the model:

$$\phi_{x_i} = \frac{df(\vec{x})}{dx_i} = \frac{f(x + h_i) - f(x - h_i)}{2h_i}. \tag{2.7}$$

Where our true model f is being differentiated by the input feature x_i for which we want to determine the feature importance. The perturbation size h_i defines how far we perturb the instance we want to explain in the direction of the feature (i) for which the feature importance is being calculated. The idea is that the most important feature at $\vec{x}$ will have the greatest gradient at $\vec{x}$. This is the approach taken to explain linear models under the assumption that the expected value for each feature is 0. How does this type of explainer compare to the more advanced explainability methods?

2.2 Duffing Oscillator

In this work we will be using the Duffing Oscillator as a basis for the system on which we test the explainers. The Duffing Oscillator [3] is a damped and driven oscillator with a non-linear stiffness. It is characterised by the equation of motion:

$$\ddot{x} + \delta\dot{x} + \alpha x + \beta x^3 = \gamma\cos(\omega t), \tag{2.8}$$

where α is the linear stiffness of the spring, β is the non-linearity in the restoring force, γ is the amplitude of a periodic driving force, δ is the damping factor and ω is the period of the driving force. The Duffing oscillator in its simplest form is just a simple harmonic oscillator ($\beta, \gamma, \delta = 0; \quad \alpha > 0$), this can be solved analytically fairly easily. For the un-damped and un-driven ($\gamma = 0, \delta = 0$) case analytic solutions also exist in the form of the Jacobi elliptic functions [16].

In the general case, the Duffing Oscillator exhibits varied behaviour even when changing only γ. For low values of γ we observe the controlled behaviour shown in Fig. 2.2.

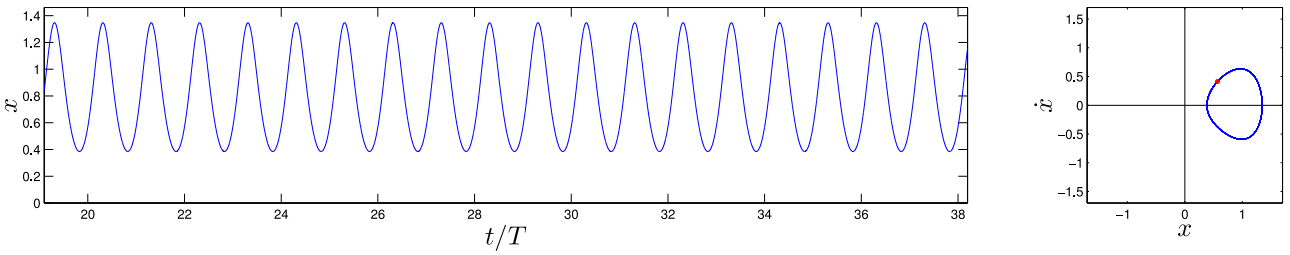


Fig. 2.2: Duffing Oscillator for $\alpha = -1$, $\beta = 1$, $\gamma = 0.2$, $\delta = 0.3$, $\omega = 1.2$. The red dots denote integer multiple of the period of oscillation [17].

The oscillations here are restricted to being around a single potential well. Increasing the driving amplitude to $\gamma = 0.37$ gives us a closed orbit about both potential wells. This is shown in Fig. 2.3.

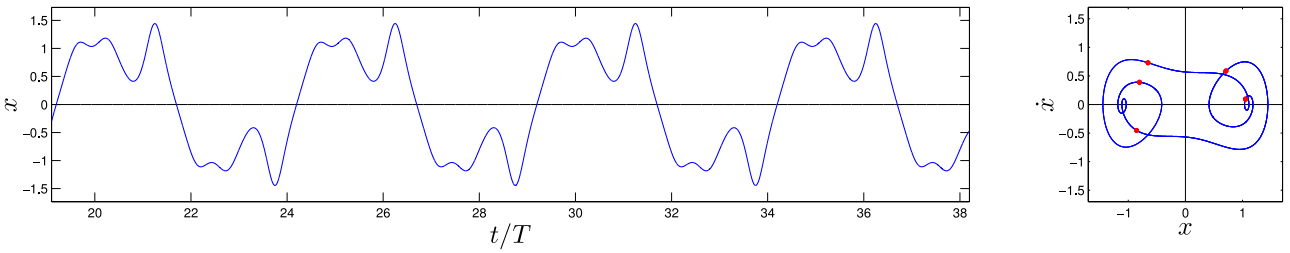


Fig. 2.3: Duffing Oscillator for $\alpha = -1$, $\beta = 1$, $\gamma = 0.37$, $\delta = 0.3$, $\omega = 1.2$. The red dots denote integer multiple of the period of oscillation [17].

The system then exhibits chaotic behaviour at $\gamma = 0.5$ (Fig. 2.4.) before returning to a predictable system for larger values of γ.

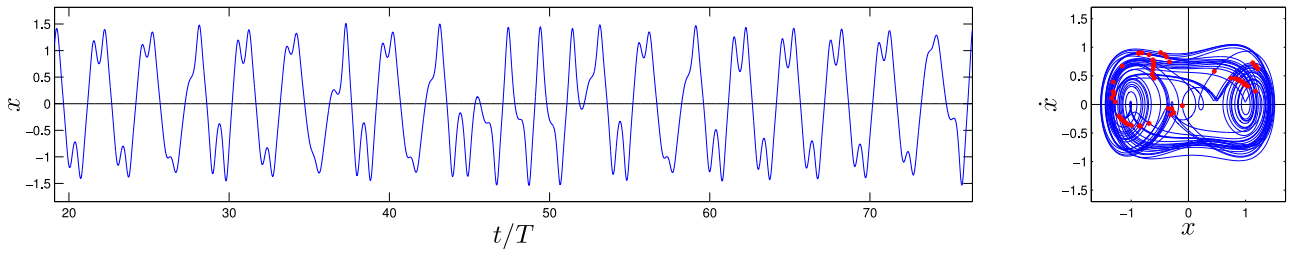


Fig. 2.4: Duffing Oscillator for $\alpha = -1$, $\beta = 1$, $\gamma = 0.5$, $\delta = 0.3$, $\omega = 1.2$. The red dots denote integer multiple of the period of oscillation [17].

From these examples, we see that the system displays varied behaviour. We can use this to construct test cases. In order to simulate the general case of the Duffing Oscillator we resort to numerical integration.

2.2.1 Numerical Integration

For the numerical integration we used SciPy's ([18]) solve_ivp. This method uses an Explicit Runge-Kutta method of order 5(4). This means that the error and accuracy are controlled using a 4th order Runge Kutta method. Steps are then taken with a 5th order Runge Kutta Method [19]. This results in a method of higher order than the alternative leapfrog method. As a result, the error of the method is smaller. The benefits of leapfrog, such as the conservation of energy and ability to be backwards integrated, do not outweigh this benefit.

2.2.2 Data Generation

Using the numerical integration we can now generate our data points. The first step in the data generation is to choose a parameter configuration for the Duffing Oscillator. Shown in Fig. 2.5. are the potential landscapes generated by certain choices of α and β.

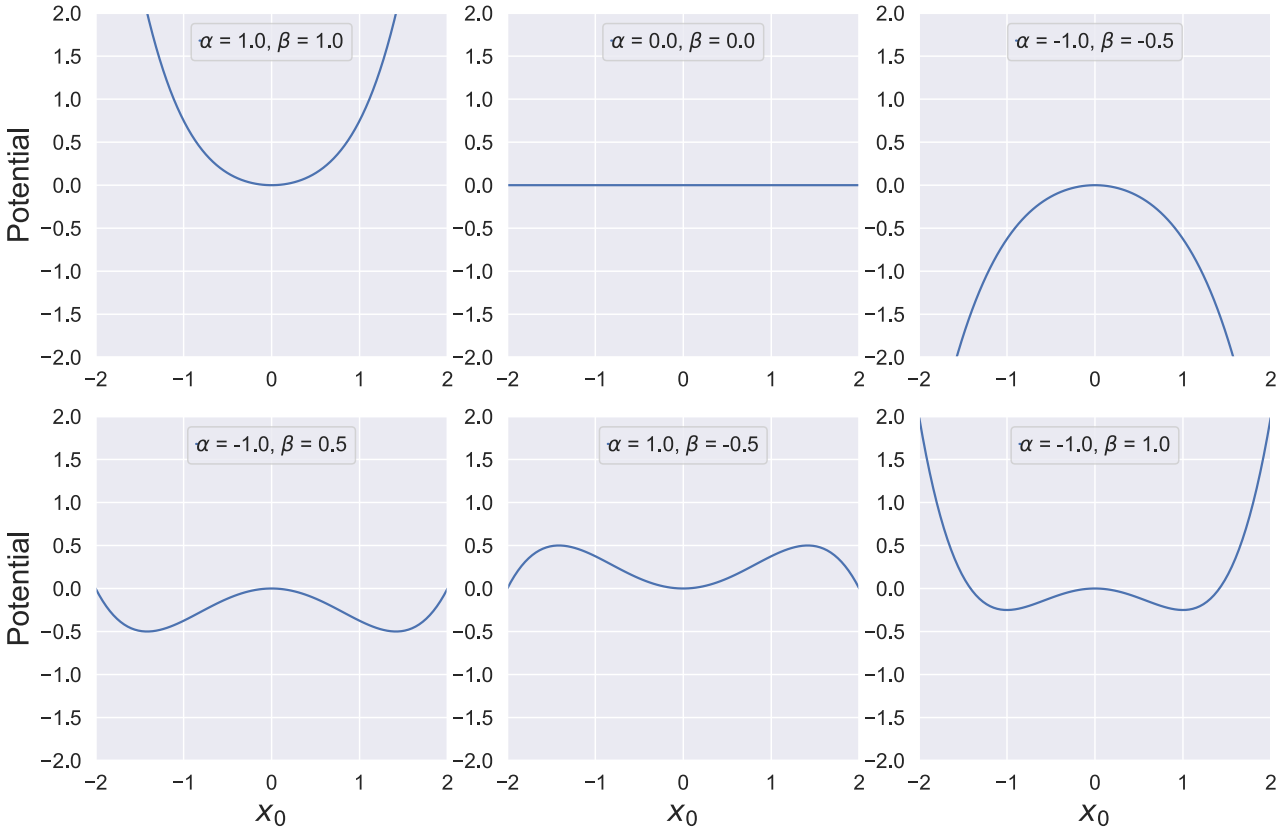


Fig. 2.5: Potentials that arise from the titular parameter configurations (α and β) of the Duffing Oscillator.

Parameter configurations with negative values of β represent variations on the potential hill. For these configurations positions far from the origin are energetically favourable. This results in the velocity (v) and

position (x) diverging for large times (t). In order to prevent this from breaking the numerical integration a stopping condition was introduced. When a trajectory exceeds the values of $|x| > 50$ or $|v| > 50$ the integration is halted and points are considered stationary at the point at which the stopping condition was triggered.

When we choose random starting points from $(x_0, v_0) \in [-2, 2] \times [-2, 2]$ and let the integration run for 100 timesteps we end up with the datasets shown in Fig. 2.6. These correspond to the potentials shown in Fig. 2.5.

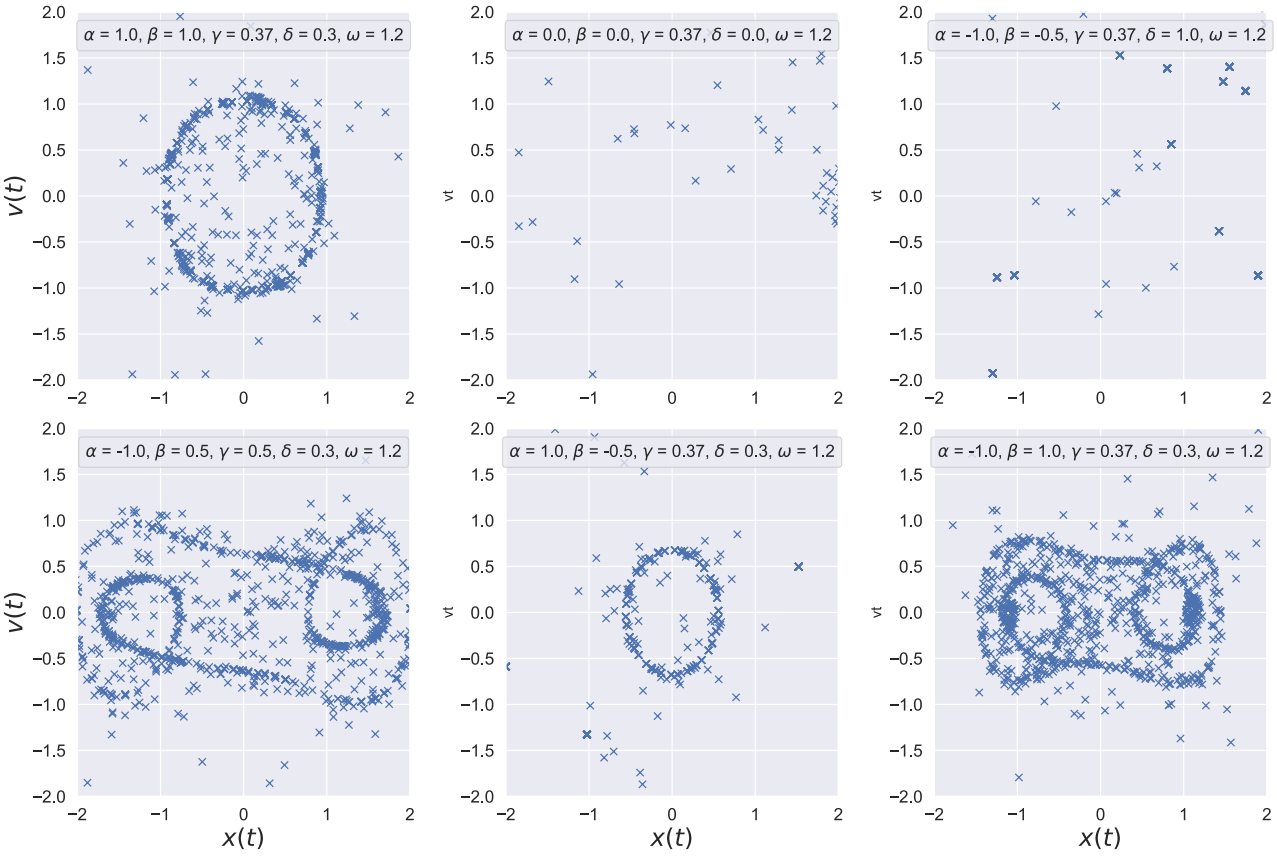


Fig. 2.6: The generated datasets (x_t, v_t) for the potentials shown in Fig. 2.5. The generating parameter configuration of each dataset is listed in the title of the plot.

What we find is that this simulation does indeed display varied behaviour based on the parameters. The behaviour varies from a chaotic trajectory at $\gamma = 0.5$ to all points oscillating within a potential well in a synchronised fashion at $\gamma = 0.2$.

2.2.3 Simulation

Since we have access to the data generation process we can create a "true model". This is a model that replicates the data generation process by repeating the Runge Kutta integration. The predictions made by this model can then be evaluated by the explainability frameworks as if it were a machine learning model. We will refer to this perfect model as "the simulation" and differences between it and a machine learning model should alert us to the indications of what caused of the machine learning model's failure to predict correctly.

2.2.4 Duffing Oscillator Parameter Configurations

Altering the values of $\alpha, \beta, \gamma, \delta$ and ω leads to a variety of systems. We will look at the concrete cases detailed below. In this section we look at two cases that are designed to elicit different responses in terms of feature importance. These are the cases that were the main focus for this work.

Double Well

The first test case was created using the configuration found on the Wikipedia page for the Duffing oscillator [17]. The potential for this configuration is shown in Fig. 2.7.

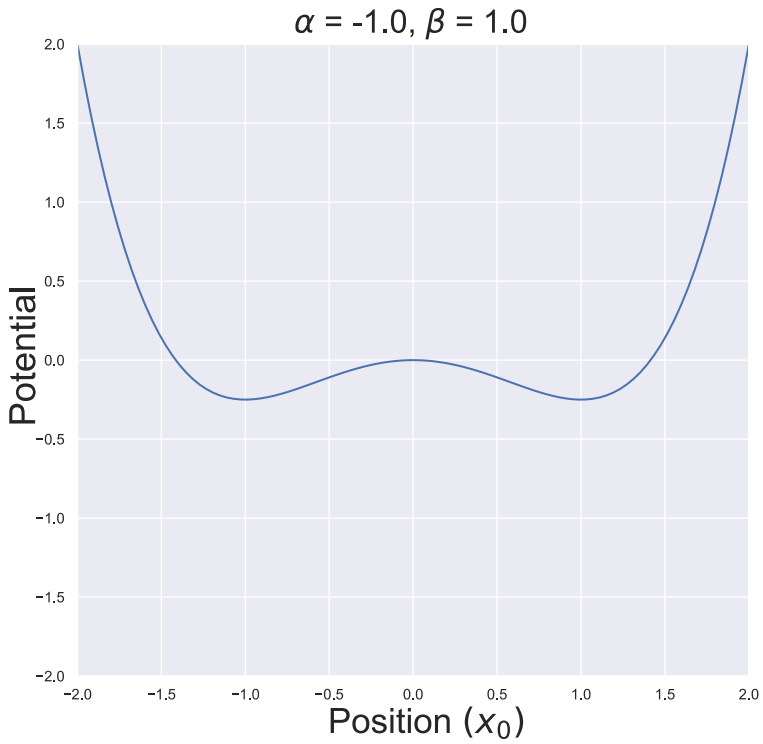


Fig. 2.7: Double well potential of the Duffing Oscillator for $\alpha = -1, \beta = 1$.

With a parameter configuration of $\alpha = -1$, $\beta = 1$, $\gamma = 0.37$, $\delta = 0.3$, $\omega = 1.2$ we get the system shown in Fig. 2.8. This system exhibits light damping.

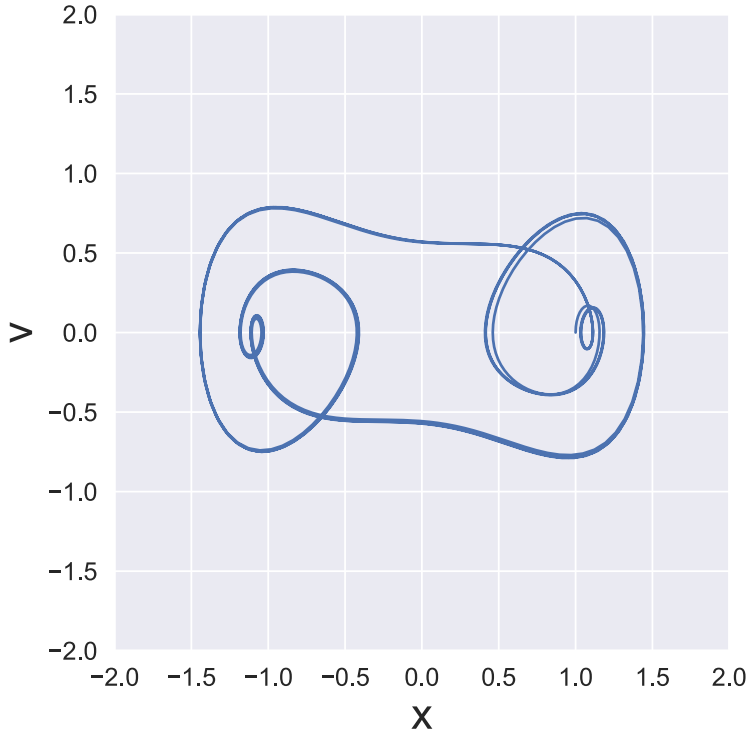


Fig. 2.8: The Duffing oscillator for a system with $(\alpha = -1, \beta = 1, \gamma = 0.37, \delta = 0.3, \omega = 1.2)$. This is the trajectory for a point starting at (-1.86, 0.30).

This figure shows the trajectory for a single initial position ($x_0 = 1.0$, $v_0 = 0.0$). To assign feature importance to the starting position and velocity we need a dataset for which these values vary. Using the data generation method described in Sec. 2.2.2., this results in the dataset shown in Fig. 2.9.

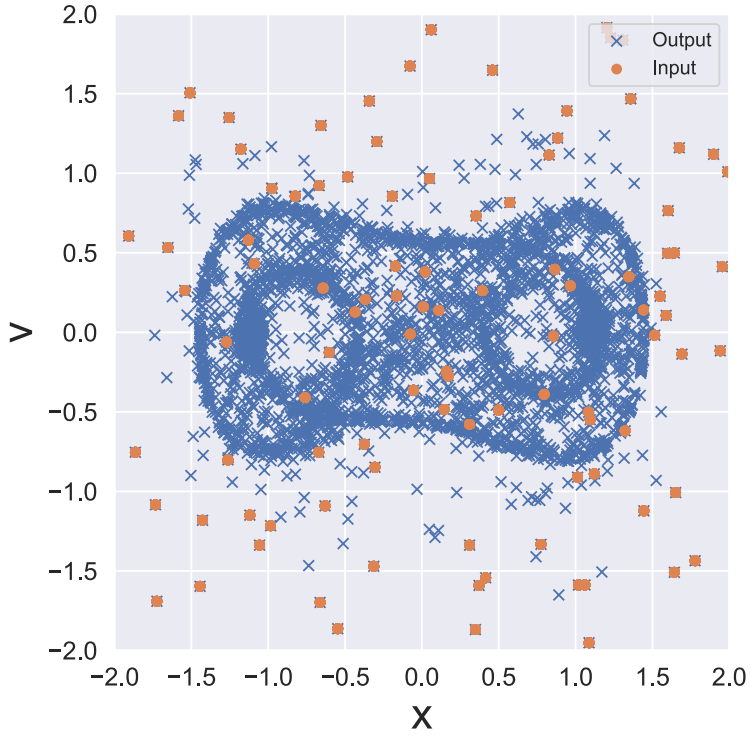


Fig. 2.9: Generated data for the lightly damped ($\alpha = -1, \beta = 1, \gamma = 0.37, \delta = 0.3, \omega = 1.2$) double well configuration. Orange: (x_0, v_0); Blue: (x_t, v_t). t is not shown and varies for each datapoint.

This test case is difficult for a human to evaluate feature importance in. In this configuration, a single closed trajectory orbits both wells. Any starting position being able to access any well means that it is difficult to say where a point may be at time t given its initial position (x_0, v_0). As such we would expect feature importance to be similar across the features.

Damped Double Well

For this reason, the next test case to consider is a more heavily damped version of the previous configuration. In this case, we set δ to 1. Shown in Fig. 2.10. is the generated dataset for this case.

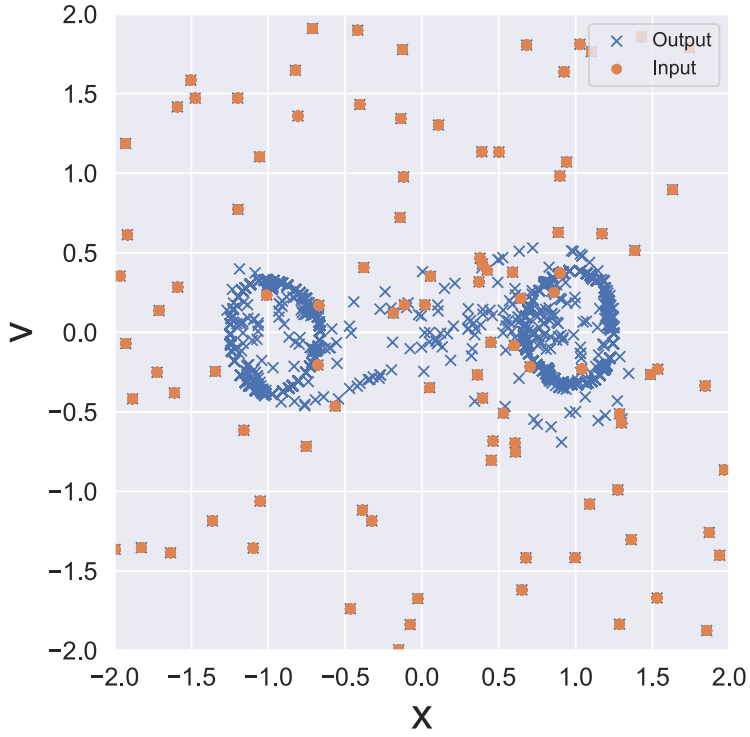


Fig. 2.10: Generated data for the heavily damped ($\alpha = -1, \beta = 1, \gamma = 0.37, \delta = 1.0, \omega = 1.2$) double well configuration. Orange: (x_0, v_0); Blue: (x_t, v_t). t is not shown and varies for each datapoint.

This dataset can be intuitively explained. Trajectories now end up in one of the two wells and cannot leave. The initial conditions determine which of the two wells a point ends up in. From this, we can construct what is almost a classification task. For points originating to the left of the origin the left well is a likely final destination. Points originating from the right of the origin end up in the right well. The initial velocity will impact points that start around the origin. When a trajectory starts around the origin with a high initial velocity this can push the point towards a well. Since trajectories do not leave the well, the time becomes irrelevant as soon as a point has entered a well. As a result, the time should have a low feature attribution for this case.

2.3 Neural Networks

Explainers are designed to evaluate the decision making of black-box machines such as neural networks. In addition to testing them on the simulation discussed in Sec. 2.2.3 we apply them to some simple neural network architectures. For this purpose, we propose two neural network architectures that should allow some contrast in results in terms of accuracy. The neural network is simply modelling our function:

$$f(x_0, v_0, t) \rightarrow (x_t, v_t) \tag{2.9}$$

To implement the neural networks used in this work, we used the TensorFlow[20] library in python. Data preprocessing was done using scikit-learn [21].

2.3.1 Data Preprocessing

All generated features were scaled to the window [0,1] to improve the fitting of the neural networks and to eliminate the scale of a feature from being a relevant factor in any of the explainability calculations. Scaling was performed using scikit-learn's MinMaxScaler.

2.3.2 Deep Neural Network

The first neural network architecture is a deep neural network (DNN) similar to the one used by Renato Belotti [4]. This neural network consists of a series of fully connected 200 neuron layers followed by a few 32 neuron layers that implement different activation functions. Despite the model's simplicity we will see that it is able to capture the physics of the Duffing Oscillator as long as we remain within a restricted space of (x_0, v_0, t).

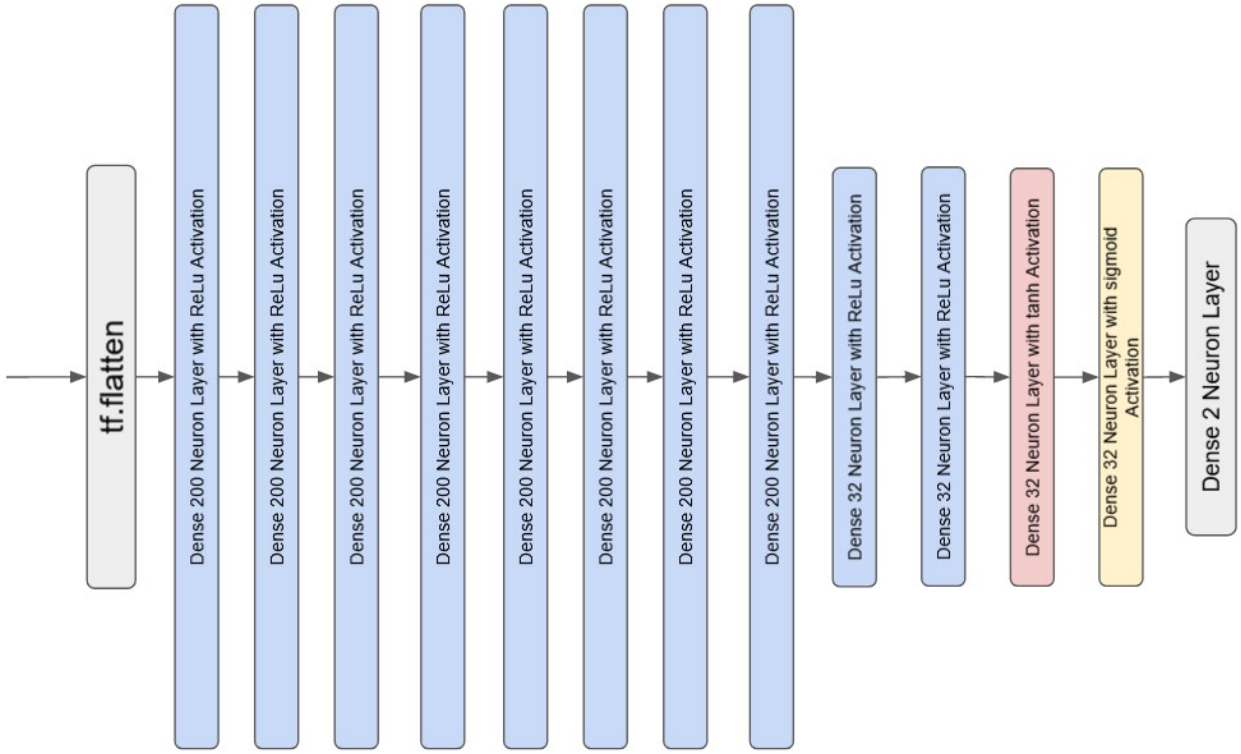


Fig. 2.11: Structure of the deep neural network used to model the Duffing Oscillator. Shown is a layer that flattens the input data, followed by 8 consecutive, dense, 200 neuron, ReLU layers and then 4 32 unit layers with the activation functions ReLU, ReLU, tanh and sigmoid. The output comes from a final dense 2 neuron layer.

As can be seen in Fig. 2.11 the network first flattens the input vector and then passes it through 8 consecutive fully connected layers. It is then fed through a series of rectified linear units (ReLU), tanh and *sigmoid* layers. This part of the architecture was inspired by [22].

2.3.3 Shallow Neural Network

This neural network is designed to be insufficient to accurately model the data. The goal is to find out whether we can use explainers to identify functioning neural networks. For this purpose, we use only a single fully connected 200 neuron layer and one fully connected 32 neuron layer. The architecture of the shallow neural network (SNN) is shown in Fig. 2.12.

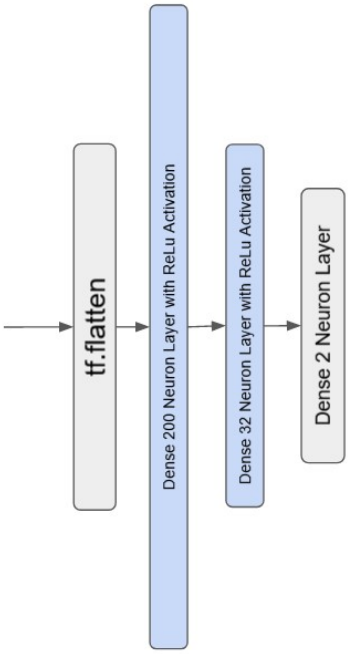


Fig. 2.12: Structure of the shallow neural network designed to insufficiently model the Duffing Oscillator. Shown is a layer that flattens the input data, followed by a single, dense, 200 neuron, ReLU neuron layer and then a 32 neuron ReLU layer. The output comes from a final dense 2 neuron layer.

Chapter 3

Methods

In this chapter we will detail the work necessary to setup the environment for testing the explainers. First we will look at tuning parameters. Then we will look at training the networks to use in evaluating the explainability methods. Since a lot of the methods used in this work rely on random sampling we facilitate the reproducibility of the results shown in this work by seeding the random number generators used in all notebooks and Python scripts.

3.1 Hyperparameter Tuning

In this section we tune our pipeline so that each step performs as desired. This includes tuning the kernel width for LIME, the gradient explainer's step size and the neural network training batch size.

3.1.1 LIME

LIME is a very flexible explainability framework as almost everything can be tuned. We can decide on how many points to sample, how to weight them and which method to use to fit them. The LIME kernel is our method of defining a neighbourhood of points of the instance x we are trying to explain.

LIME Kernel

The kernel we use for LIME is a simple Gaussian kernel:

$$\ker(d) = e^{\frac{-d^2}{h^2}} \tag{3.1}$$

Where d is the euclidean distance to the explained instance and h is the kernel width. However, this choice of kernel is rather arbitrary and we could use another. As a result this is an un-tuned parameter of LIME. As discussed in [23], properly tuning LIME requires having an expert attribute true feature importances to the model. These can then be used to choose the parameters for the explainer. This is the main limitation of the method.

LIME Kernel Width

In order to tune the kernel width we look at the impact it has on feature importance for our system. In Fig. 3.1. is shown the feature importance for a random set of points in our heavily damped two well system for varying kernel widths.

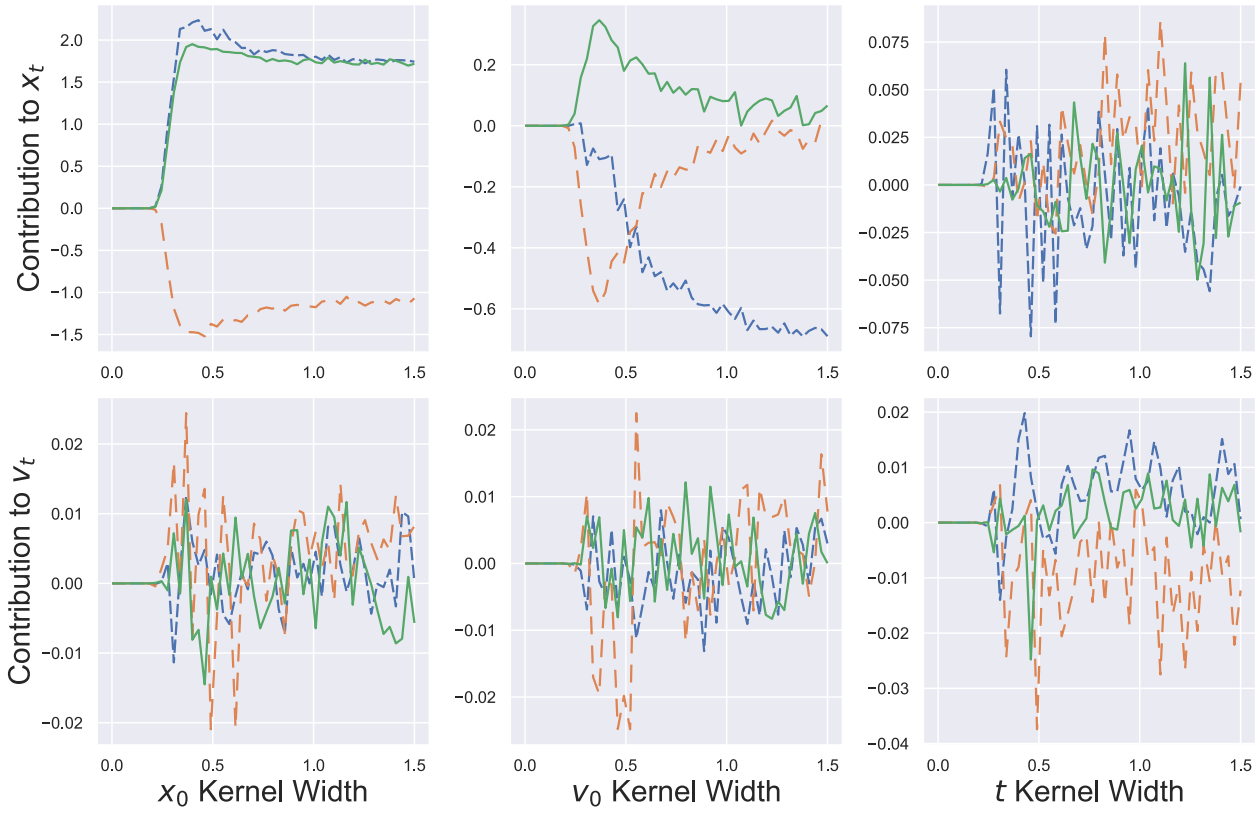


Fig. 3.1: LIME kernel width convergence study. Shown is the dependence of the feature attribution of x_0 to x_t for 3 random data points at different kernel widths for the heavily damped two well system. LIME is being used with discretised feature values. Each data point corresponds to a different colour.

From the image we can see that the feature importance begins at 0 for each point before jumping to a value around which it oscillates at a kernel width of around 0.25. This behaviour is consistent across all features. Some of this behaviour is likely the result of noise [24]. Another contribution is the fact that, with our configuration of LIME, feature values are discretised into quartiles [12]. Since our data is scaled to $[0, 1]$ (Sec. 2.3.1.), this results in kernel widths up to 0.25 delivering a non-zero feature importance. As we increase the kernel width we get to a threshold where other feature values are taken into account when training the surrogate model. The alternative is to not discretise the data, this results in the feature importance changing with the kernel width as shown in Fig. 3.2.,

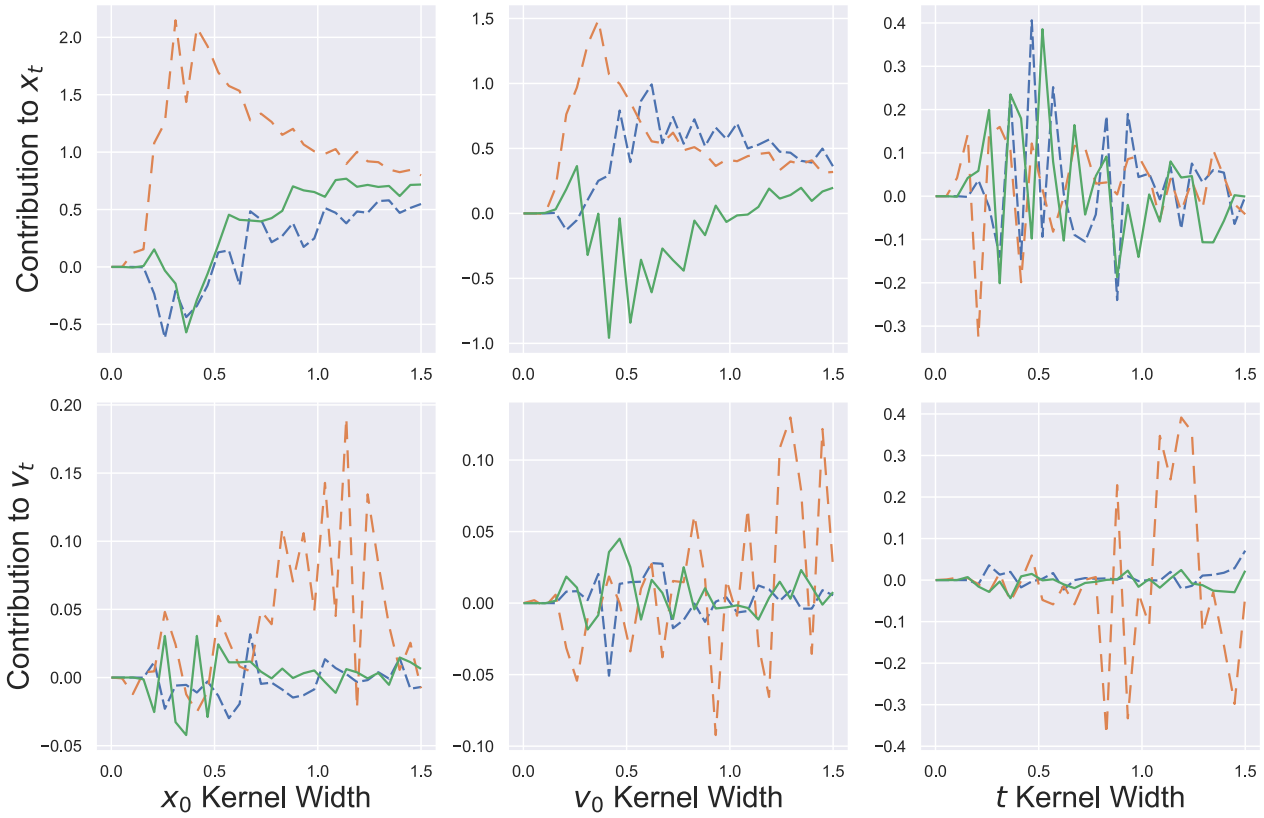


Fig. 3.2: LIME kernel width convergence study. Shown is the dependence of the feature attribution of x_0 to x_t for 3 random data points at different kernel widths for the heavily damped two well system. LIME is being used with continuous feature values. Each data point corresponds to a different colour.

where we see the feature importance assume non-zero values for values of the kernel width of around 0.10. For the continuous feature values LIME never attributes a constant feature importance for a range of kernel widths unlike LIME with discretised values.

We will be using LIME with discretised values with the default value for the kernel width of $0.75 \times \#$ features. Which is between 1.2 and 1.5 for the different configurations. These values for the kernel width places us outside the region where not enough points are available to train a model. Testing suggested that these kernel widths delivered values similar to SHAP. The contributions of v_0 to x_t change drastically for different kernel widths. This is the first indication that LIME is sensitive to the kernel width and that this may lead to problems later on.

Increasing the number of samples taken for LIME smooths out this curve as we get more training points at each kernel width, leading to a more consistent surrogate model. The benefits of the more consistent results are outweighed by the increased computational runtime of LIME. The runtime increases drastically for the simulation as we need to pass each of these samples through the model in order to use it as a training point. As such, we have used 500 samples to train the surrogate model at each point.

3.1.2 Gradient Step Size

When taking the gradient for the numerical explainer (Sec. 2.1.5.) we need to make sure the step size is chosen correctly. In order to determine the best step size (h_i) for the derivative we perform a convergence study on the numerical gradient calculation. The results for three random datapoints are shown in Fig. 3.3.

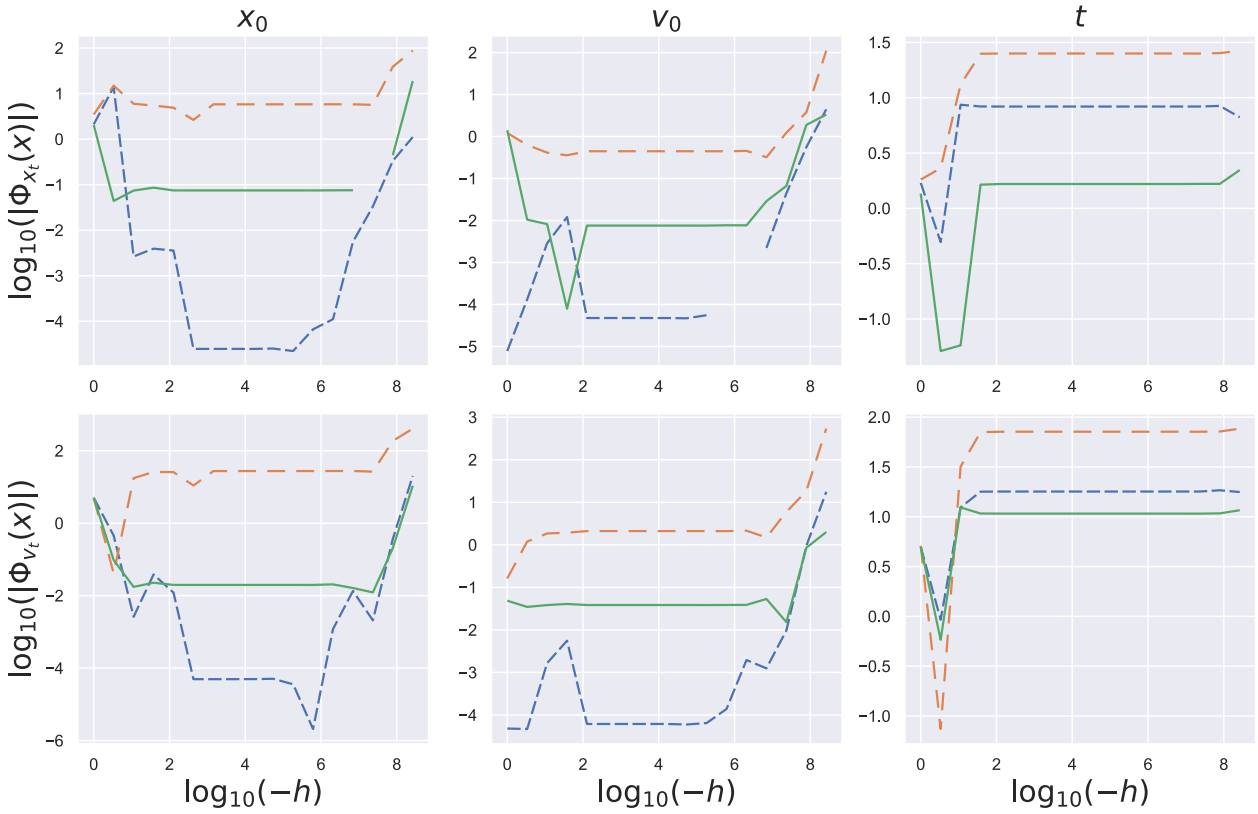


Fig. 3.3: Convergence study for the central differences gradient calculation. Plotted is the log_{10} of the absolute value of the gradient against the log_{10} of the negative step size for three randomly determined datapoints. Each data point corresponds to a different colour.

As can be seen from the figure, the gradient changes drastically when the step size is large, this is because we start exploring too far from the point at which we are taking the gradient and are no longer looking at the problem locally. When we make the step size too small the explainer also struggles as we are now within the error bounds of the data generation method. As such we chose a step size of 10^{-3} for the numerical gradient calculation which is within the constant region for all six plots.

3.1.3 Neural Networks

Neural Network Performance is improved by parameter tuning. We tuned the training batch size, learning rate and early stopping conditions. We used the Adam Optimiser [25], with a learning rate of 0.001 and an exponential decay rate for the 1st moment estimates of $\beta_1 = 0.7$. These parameters were determined by trial and error.

Batch Size

The networks were trained with variable batch sizes and a batch size of 1024 was found to work well. In Fig. 3.4. we see the model training and validation loss for different batch sizes.

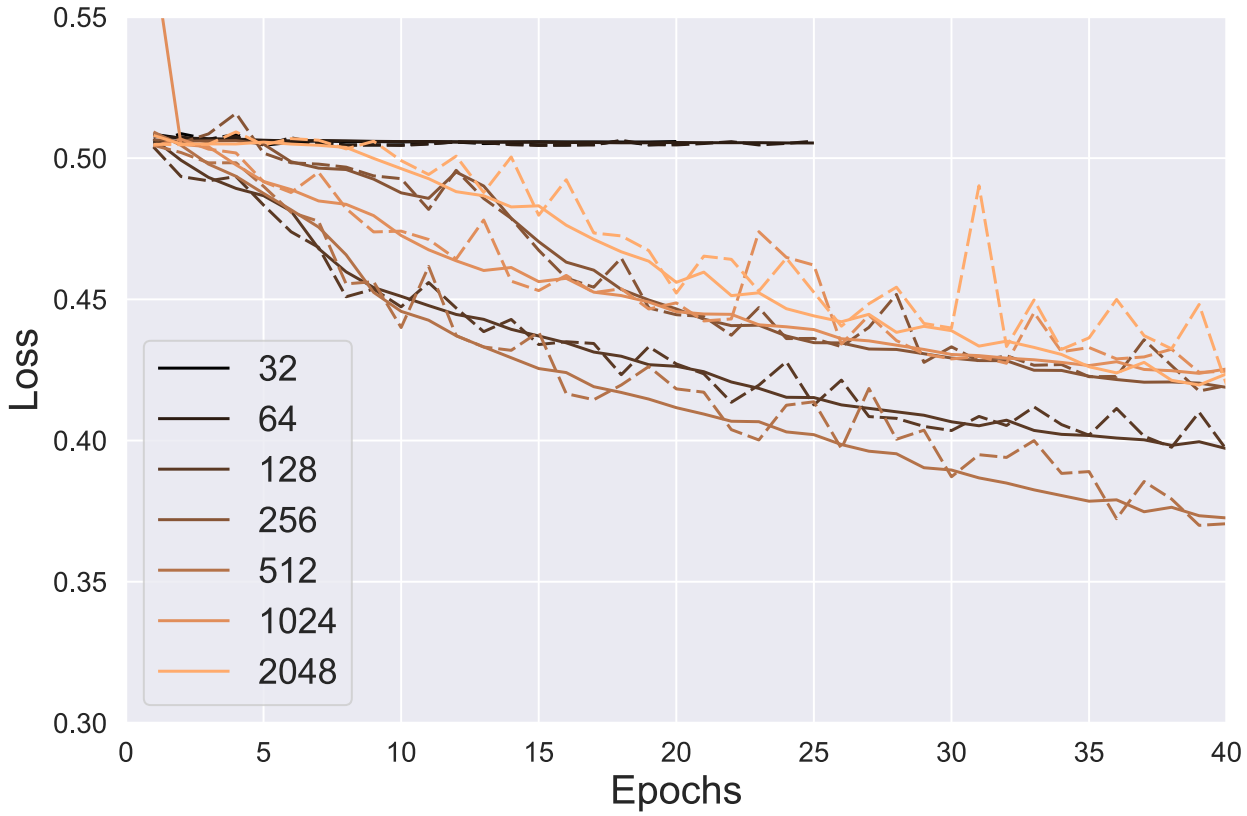


Fig. 3.4: Network training (continuous) and validation (dashed) loss for different batch sizes. Training is performed on the dataset for the lightly damped double well with 100,000 datapoints.

From the figure we conclude that the network struggles with training when we keep the batch size too low. For batch sizes under 128 the model doesn't train at all. Moreover, training is substantially slower in the case of low batch sizes as each epoch requires a greater number of iterations. Since we are working with a relatively low dimensional space (of order 10^1) when compared to a typical image processing model (order 10^6+) we do not need to worry about memory issues whilst training the network. As such we chose a batch size of 1024 for our model training.

Overfitting

In order to combat overfitting the model uses a validation set to measure model performance. For this purpose 20% of the training data is set aside in order to validate the model. The model is then supplied with an early stopping condition that requires improvement in the validation loss in order to keep training. We found that, since the model fluctuated, stopping model training after 3 epochs without improvement resulted in reliable network performance.

3.2 Network Training

Now we get onto training the models on which we evaluate the explainability methods. We will look at three different datasets created by different configurations of α, β, γ, δ and ω.

3.2.1 Lightly Damped Double Well

In this section we train the networks on the lightly damped double well. We define this as the system defined by the parameter configuration $\alpha = -1$, $\beta = 1$, $\gamma = 0.37$, $\delta = 0.3$, $\omega = 1.2$. In this section we will be focusing on the setup in which we model the system:

$$\begin{aligned} &f : \mathbb{R}^2 \times \mathbb{R} \to \mathbb{R}^2, \\ &f(x_0, v_0, t) \to (x_t, v_t). \end{aligned} \tag{3.2}$$

Where x_t, v_t are the position and velocity of a point in a Duffing oscillator trajectory originating at x_0, v_0 at a time t. For the systems concerning additional input variables, the neural networks are not discussed but training metrics can be found in App. A.

Network Performance

In order to evaluate model performance we look at the metrics of network training and validation loss, and prediction error. We define the prediction error as:

$$error = \frac{||f(x) - \hat{f}(x)||}{|f(x)|}. \tag{3.3}$$

Where the difference between the network prediction ($\hat{f}(x)$) and the actual value of the label ($f(x)$) is taken and normalised by the value of the label. These metrics are depicted, alongside the potential, in Fig. 3.5.

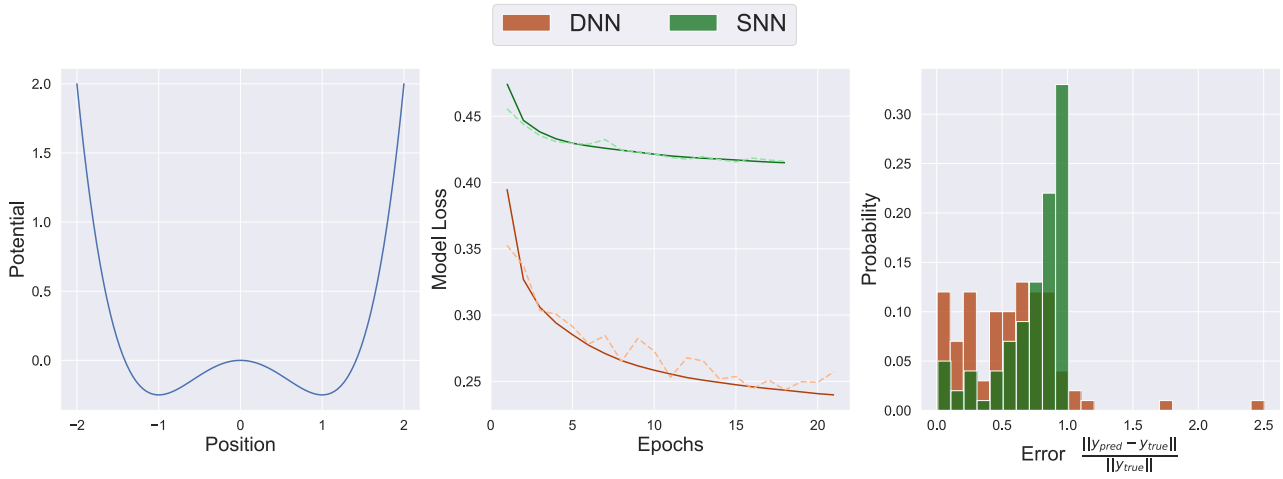


Fig. 3.5: The model training and validation accuracy for the shallow (SNN) and deep (DNN) neural networks. Left: Generating Potential defined by the parameter configuration ($\alpha = -1$, $\beta = 1$, $\gamma = 0.37$, $\delta = 0.3$, $\omega = 1.2$); Centre: Model training (continuous) and validation (dashed) loss for the SNN and DNN; Right: Error plot that compares the error on the validation set for the two models. Outliers, the largest 5% of the error, are removed.

For the error plot, the largest 5% of error values were not shown. This was done to ignore outliers. Outliers crop up for small values of $|f(x)|$. This can lead to very large errors for a few datapoints. These can be due to the error in the simulation and should not be used to judge the machine learning model. From the figure we can deduce that the deep neural network performs significantly better at modelling the data than the shallow neural network. Neither network performs particularly well. Despite the superior performance of the DNN over the SNN, it has a mean error of ≈ 0.6. We verify that the networks are replicating the simulation by looking at the network predictions.

Network Predictions

In order to best gauge the network performance and where it may be going wrong, a plot of the network predictions is shown in Fig. 3.6.

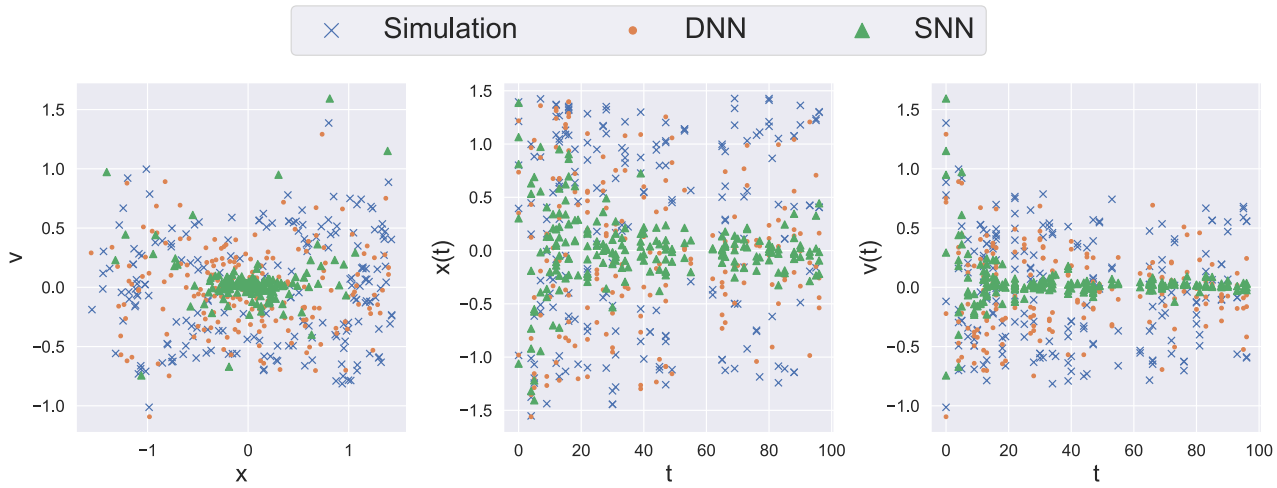


Fig. 3.6: Predictions made by the machine learning networks for the lightly damped double well potential ($\alpha = -1$, $\beta = 1$, $\gamma = 0.37$, $\delta = 0.3$, $\omega = 1.2$). Left: Plot of the network outputs. Centre: Plot of $x(t)$ by t. Right: Plot of $v(t)$ by t.

These predictions were performed on a random subset of the validation set. From the figure we can see that the DNN is better at replicating the data than the SNN. It is also roughly aligned with the simulation in terms of the spread of the datapoints. It does still struggle with replicating the exact behaviour. The SNN places its predictions in the centre of mass of the training labels. This means that it performs roughly equally badly for any point that we pass through the model. This leads to the large peak in the prediction error at $error \approx 1.0$.

3.2.2 Heavily Damped Double Well

Now we move on to a system where we expect model performance to be better. We increase the damping to $\delta = 1$. This system is characterised by the parameter configuration $\alpha = -1$, $\beta = 1$, $\gamma = 0.37$, $\delta = 0.3$, $\omega = 1.2$. This heavy damping ensures that points that enter a potential well are no longer able to leave it. Points now oscillate around the minimum of the potential well with an amplitude dependent on the amplitude of the driving force.

Network Performance

Looking, once again at the network performance. Shown in Fig. 3.7. is the performance plot for this configuration in the setup 3.2.

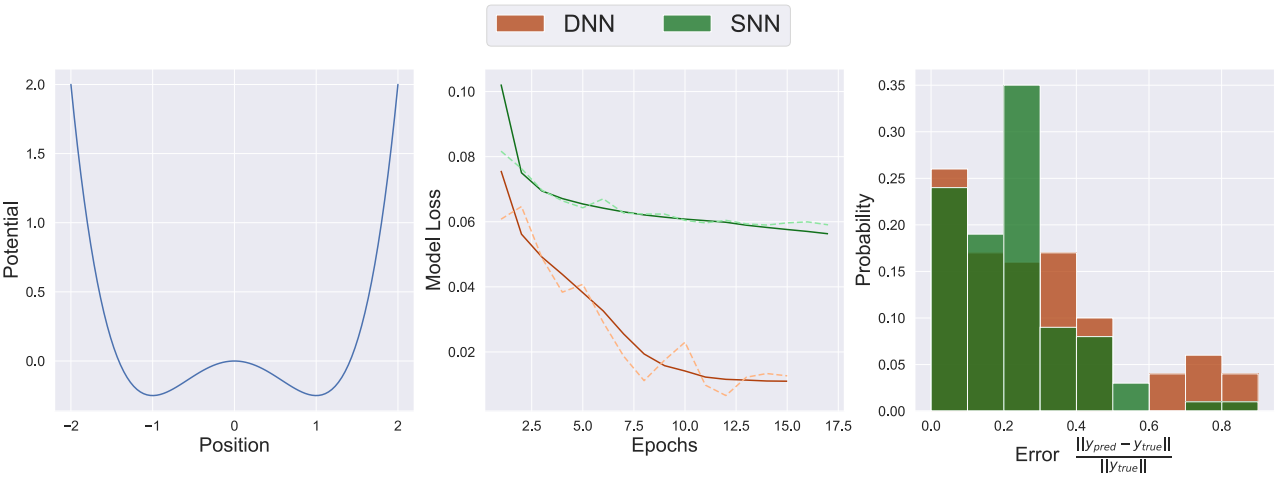


Fig. 3.7: Network training and validation accuracy for the shallow (SNN) and deep (DNN) neural networks. Left: Generating Potential ($\alpha = -1$, $\beta = 1$, $\gamma = 0.37$, $\delta = 0.3$, $\omega = 1.2$); Centre: Network training (continuous) and validation (dashed) loss for the SNN and DNN; Right: Error plot that compares the error on the validation set for the two models. Outliers, the largest 5% of the error, are removed.

This figure shows that both networks perform better than in the lightly damped case, the mean error for the DNN has decreased to ≈ 0.34. The DNN is still performing better than the SNN. The prediction error for the SNN has a significant spike at error ≈ 0.3. This spike could indicate that the model is once again putting points in the centre of mass of the training labels. We verify this by looking at the network predictions.

Network Predictions

In Fig. 3.8. we plot the predictions made by the two machine learning networks.

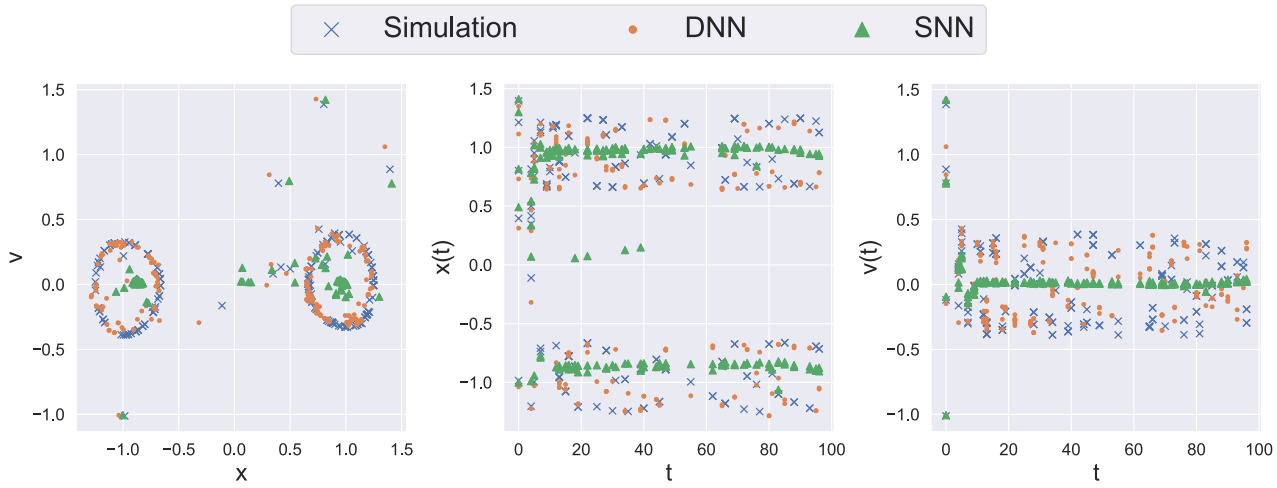


Fig. 3.8: Predictions made by the machine learning networks for the damped double well potential ($\alpha = -1$, $\beta = 1$, $\gamma = 0.37$, $\delta = 0.3$, $\omega = 1.2$); Left: Plot of the model outputs; Centre: Plot of $x(t)$ by t. Right: Plot of $v(t)$ by t.

As before these predictions were performed on a random subset of the validation set. The figure implies that the DNN is able to replicate the oscillatory behaviour of the system. This was verified by plotting the predictions for each time step and comparing the behaviour to the simulation. The SNN repeats its performance for the lightly damped case. Points travel towards one of the two potential wells and then stop moving after around 5 time-steps. Points that have not reached a potential well also stop, regardless of position. However, even the SNN is able to correctly identify the well that points end up in, as shown by the relatively small error in the predictions. A misclassification would result in an error of around 2.

3.2.3 Single Well

The last configuration we look at here is the single potential well. Where all points end up oscillating in step around the well.

Model Performance

Looking at the model performance, shown in Fig. 3.9. is the network performance plot for this parameter configuration.

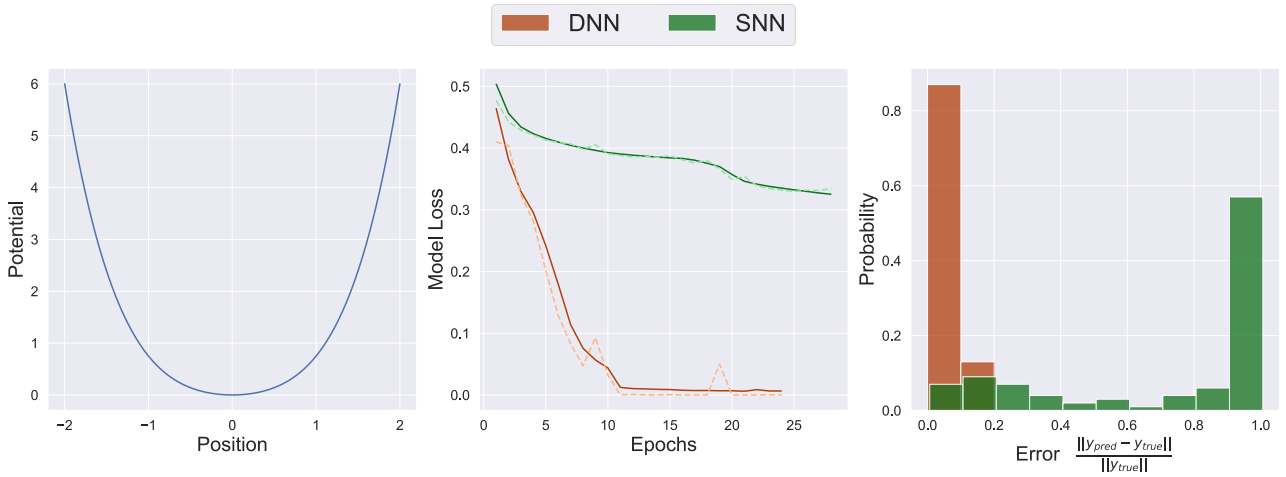


Fig. 3.9: Network training and validation accuracy for the shallow (SNN) and deep (DNN) neural networks. Left: Generating Potential ($\alpha = 1$, $\beta = 1$, $\gamma = 0.37$, $\delta = 0.3$, $\omega = 1.2$); Centre: Network training (continuous) and validation (dashed) loss for the SNN and DNN; Right: Error plot that compares the error on the validation set for the two models. Outliers, the largest 5% of the error, are removed.

The DNN performs very well here, with the average error now being ≈ 0.11. The SNN has a drop in performance when compared to the heavily damped double well. This drop in performance is unexpected, in order to explain it we now look at the predictions themselves.

Model Predictions

To understand why the simple model's error is so large we look at the predictions shown in Fig. 3.10.

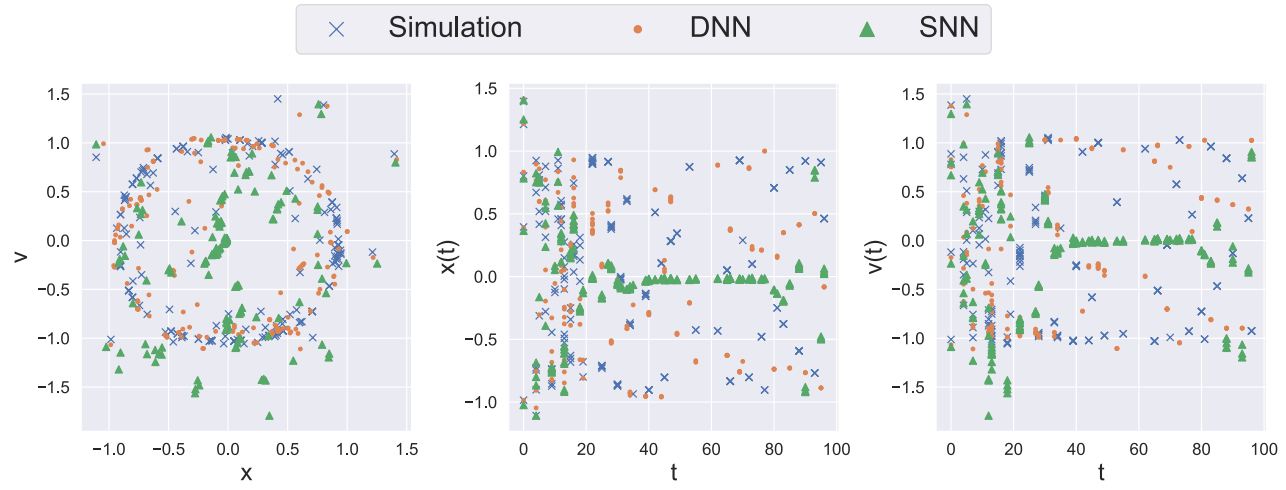


Fig. 3.10: Predictions made by the machine learning networks for the single well potential ($\alpha = 1$, $\beta = 1$, $\gamma = 0.37$, $\delta = 0.3$, $\omega = 1.2$); Left: Plot of the model outputs; Centre: Plot of $x(t)$ by t. Right: Plot of $v(t)$ by t.

The DNN's predictions align almost exactly with the actual values. The SNN's behaviour is very poor at replicating the actual values. Once again it places its predictions mostly at the centre of the well.

Chapter 4

Results

In this chapter we apply the explainability techniques to the Duffing Oscillator. We will start by looking at a system where the models use only relevant features in order to predict the trajectory for a later time t. We will then continue by including other features in order to test their impact on the explainability assigned by SHAP and LIME.

4.1 Using only (x_0, v_0, t) as Training Features

First we look at the system we focused on in Sec. 3.2. The goal is to use only the features that we would actually use when generating the data as input features for our model. In our example this means that we pass the model a vector (x_0, v_0, t). It is then tasked with generating the labels (x_t, v_t). In this case we know that each feature is contributing to the final position. The explainers are tasked with finding out which feature contributes the most. The function that the networks are modelling here is:

$$\begin{aligned} &f : \mathbb{R}^2 \times \mathbb{R} \to \mathbb{R}^2, \\ &f(x_0, v_0, t) \to (x_t, v_t). \end{aligned} \tag{4.1}$$

4.1.1 Lightly Damped Double-Well

First, we apply SHAP and LIME to the double-well with light damping, this is characterised by the parameter configuration: $\alpha = -1$, $\beta = 1$, $\gamma = 0.37$, $\delta = 0.3$, $\omega = 1.2$. In setting up this system we struggled with assigning an intuitive feature importance. We now apply the explainability frameworks to this system to see whether they can formulate intuitive feature contributions. We look at the simulation first. Doing so gives us the aggregate feature importance plotted in Fig. 4.1.

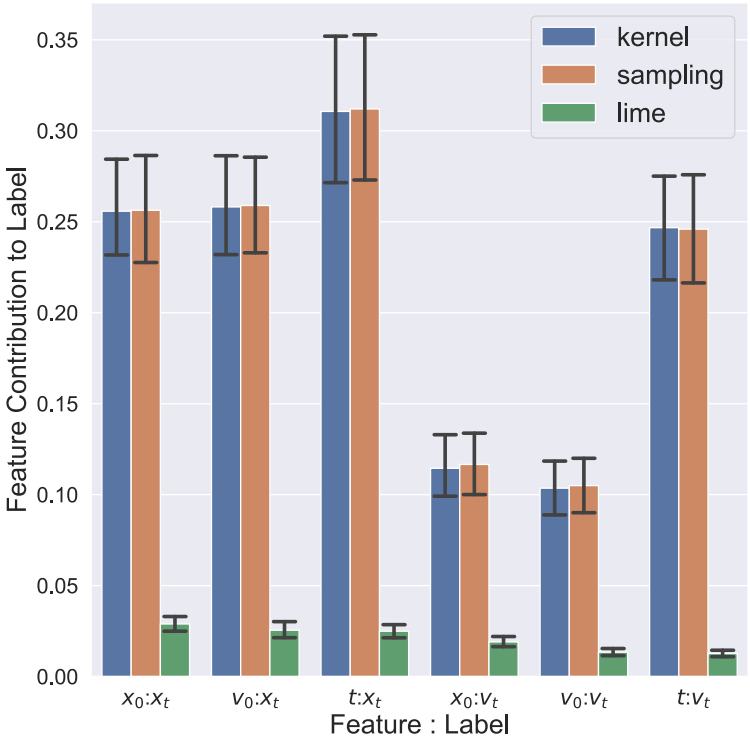


Fig. 4.1: Aggregated feature attributions for the lightly damped double potential well ($\alpha = -1$, $\beta = 1$, $\gamma = 0.37$, $\delta = 0.3$, $\omega = 1.2$). Explainability methods LIME, SHAP (kernel) and Shapley (sampling) values are applied to the simulation. Data aggregation is performed by taking the mean of the absolute values. Plotted in black are the 95% confidence intervals, determined by bootstrapping.

We plot values for three explainers, the sampling explainer is an implementation of the Shapley values that requires sampling many times to calculate feature attributions, this was superseded by kernel SHAP, which functions similarly to LIME and is discussed in Sec. 2.1. In this figure, the data aggregation is performed by taking the mean of the absolute values of the individual feature attributions. The aggregation is performed over 100 values. A 95% confidence interval is plotted, this interval is determined by the bootstrap method [26]. Looking first at SHAP this confidence interval shows us that t is consistently the most important feature in determining x_t whilst x_0 and v_0 are interchangeable. The variable t is the dominating feature in contribution to v_t. For LIME, the values are significantly lower than for SHAP and imply that x_0 contributes most to v_t. All features having roughly equal aggregate feature contributions to x_t alludes to the fact that this is a configuration where trajectories are able to transit wells. This means that we end up with the initial positions contributing equally. To try to explain why the time is contributing so heavily to v_t according to SHAP, we look at the individual feature attributions shown in Fig. 4.2.,

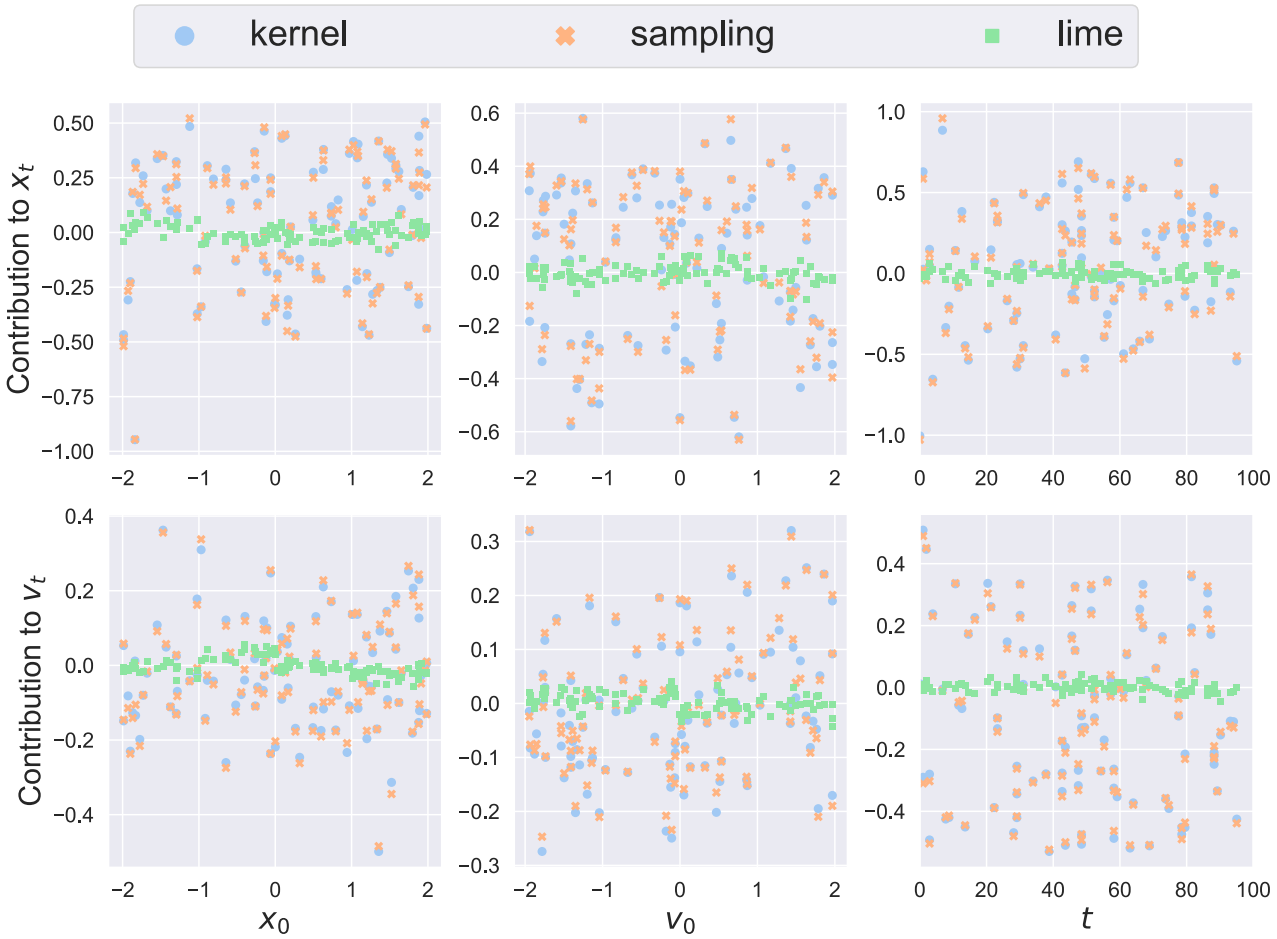


Fig. 4.2: Individual feature attributions for the lightly damped double potential well ($\alpha = -1$, $\beta = 1$, $\gamma = 0.37$, $\delta = 0.3$, $\omega = 1.2$). Plotted are the feature contributions against the corresponding feature for each of the three explainer types.

where the individual feature attribution for each feature's contribution to each label is plotted individually against the value of that feature in the input. From this, we see that there is no apparent trend in the data, with the exception of low times contributing heavily. At this time the system still has close to its initial energy, meaning that it has not been damped to the extent where points are bound to the trajectory around the two wells. For these low times, large changes in the position can occur for small changes of the time, leading to a high feature attribution to t for both x_t and v_t. The high contribution of t to both x_t and v_t makes sense as we have a system that is driven by a time-dependent force.

Looking only at the plot of the contribution against the feature itself gives an incomplete picture. In Fig. 4.3. is shown a heatmap of the feature attribution. This allows us to look at the change in the attribution over two dimensions, namely the initial position x_0 and the initial velocity v_0.

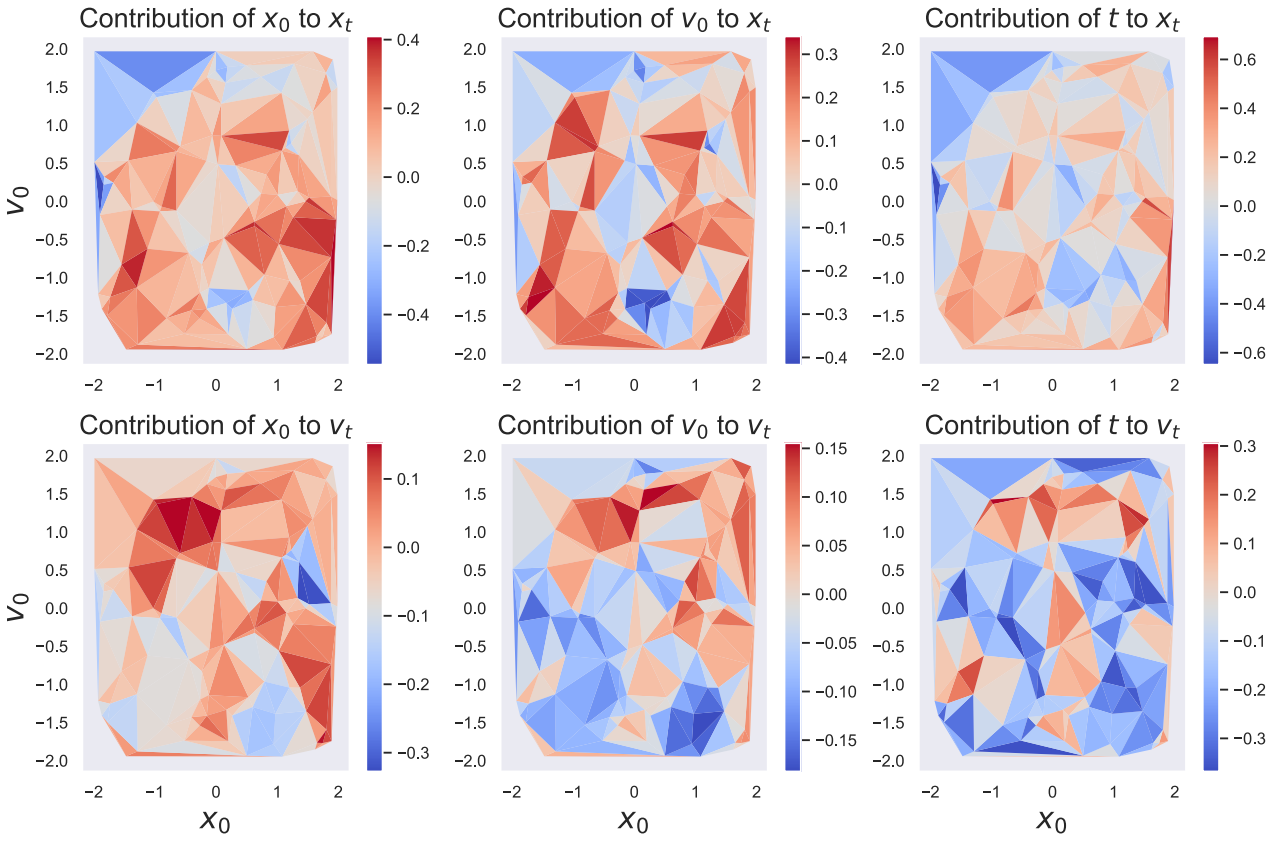


Fig. 4.3: Heatmap of the individual feature attributions for the lightly damped double potential well ($\alpha = -1$, $\beta = 1$, $\gamma = 0.37$, $\delta = 0.3$, $\omega = 1.2$). The colour at a point represents the titular explainability value at the position (x_0, v_0).

This heatmap tells us that there is no real trend in the feature attributions for varying x_0 and v_0. This lightly damped configuration doesn't tell us much from an explainability standpoint. The explainers are unable to extract clear feature importance that we are able to interpret. We also do not have a hypothesis as to what the ground truth feature importance is. This means that comparing values to the network values or the gradient explainer will not yield useful results. What we do get from these plots is the increased importance of t for low times.

4.1.2 Heavily Damped Double-Well

Instead, we increase the damping in the system, giving us the parameter configuration: $\alpha = -1$, $\beta = 1$, $\gamma = 0.37$, $\delta = 1.0$, $\omega = 1.2$. In setting up this system we postulated that the initial position should be very important here as points should end in the well they started closest to. Fig. 4.4. shows the aggregated feature importance for this system.

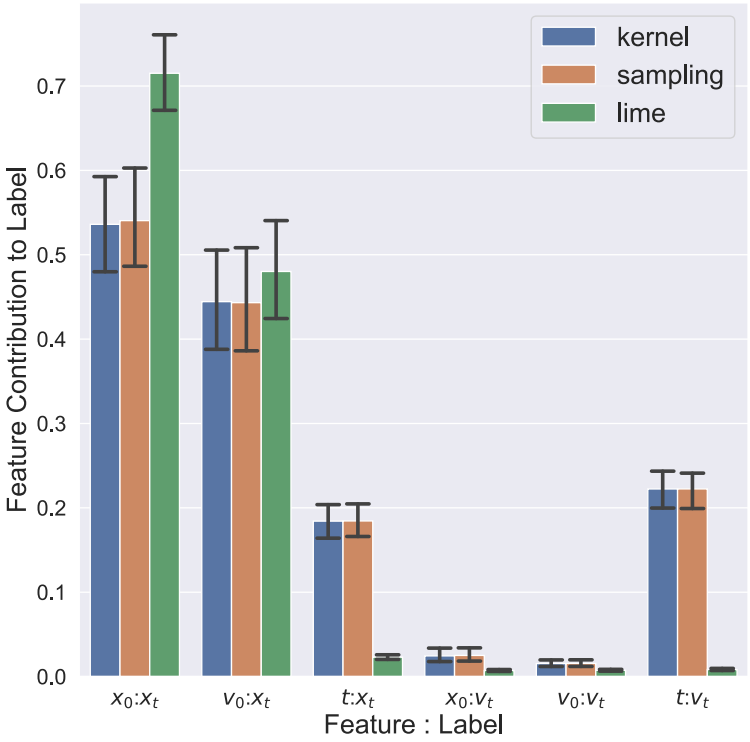


Fig. 4.4: Aggregated feature attributions for the heavily damped double potential well ($\alpha = -1$, $\beta = 1$, $\gamma = 0.37$, $\delta = 1.0$, $\omega = 1.2$). Explainability methods LIME, SHAP (kernel) and Shapley (sampling) values are applied to the simulation. Data aggregation is performed by taking the mean of the absolute values. Plotted in black are the 95% confidence intervals, determined by bootstrapping.

The aggregated feature importance shows us that the initial position (x_0) and velocity (v_0) are indeed the most important factors in predicting x_t. The velocity contributes as a high initial velocity changes the effective initial position. Starting closer to the left potential well with a strong positive velocity can move a point close enough to the right well that that ends up being its final position. For v_t only the time contributes, this is intuitive as we are looking at the system for a relatively long time span, the initial velocity is damped out within the first few steps and then the velocity depends only upon the driving force, which forces the oscillation around the potential well. This is time-dependent and thus the time determines the velocity at time t.

This is a system in which we may find some trends when looking at the individual feature attributions (Fig. 4.5.).

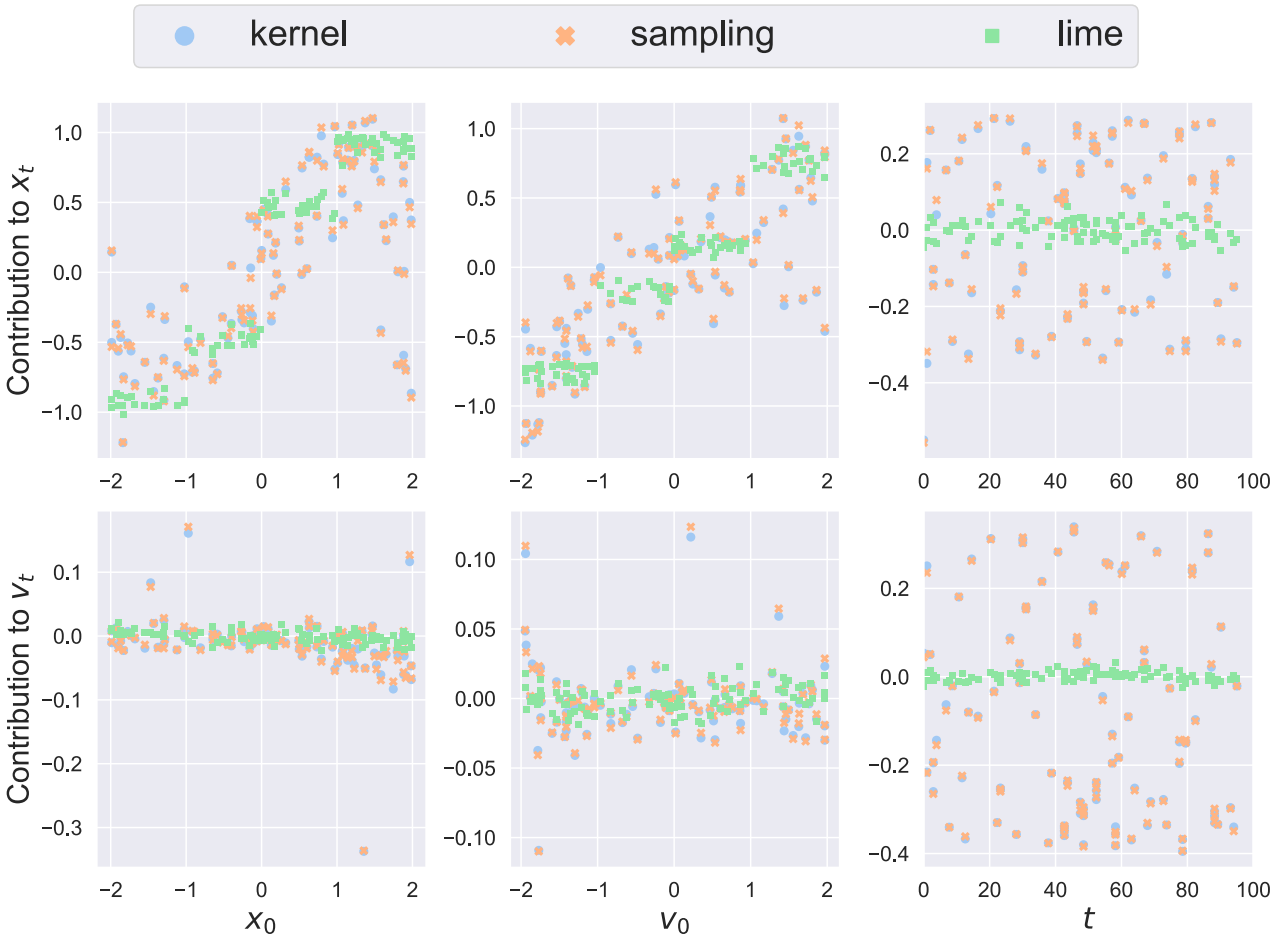


Fig. 4.5: Individual feature attributions for the heavily damped double potential well system ($\alpha = -1$, $\beta = 1$, $\gamma = 0.37$, $\delta = 1.0$, $\omega = 1.2$) with the simulation.

This figure shows that the initial position and velocity contribute linearly to the final position. This lines up with our hypothesis about feature attributions in this system and can be marked as a success of the explainers. We also observe that the contribution by x_t switches for values of $x_0 \approx 2.0$. One possible explanation for this is that points here have such high initial potential energy that they are able to transit wells despite the damping. This means that they land in the well at $x = -1$ instead of the one at $x = 1$. This well corresponds to the negative feature attribution, leading to this result. We see a similar trend for negative x_0. For both SHAP and the Shapley values explainer the attributions to t display oscillatory behaviour, this is highlighted in Fig. 4.6.

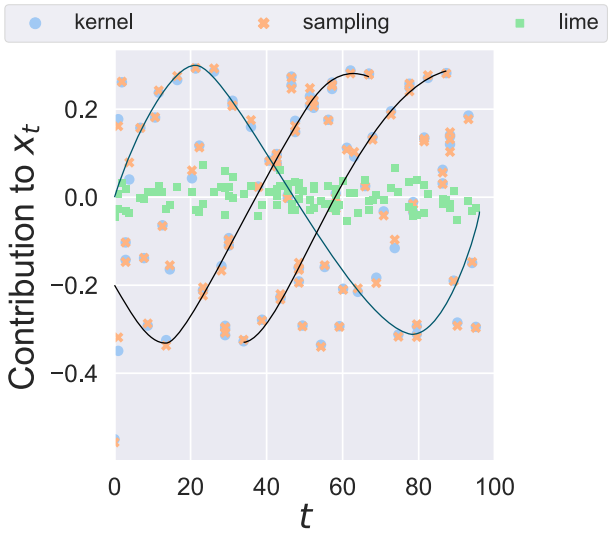


Fig. 4.6: Feature contribution of t to x_t for the heavily damped double potential well ($\alpha = -1$, $\beta = 1$, $\gamma = 0.37$, $\delta = 1.0$, $\omega = 1.2$). Highlighted (dark continuous lines) is the oscillatory behaviour of the feature contribution of t to x_t.

This shows that the explainer is capturing the oscillatory behaviour of the system. The period of this oscillation does not align with that of the Duffing Oscillator. One possible explanation for this is that the relatively small number of background samples and large spacing between the generated time steps leads to aliasing. To test this, we perform an identical experiment, changing only the number of time steps in the simulation. In order to get a higher temporal resolution we use an end time of 20 instead of 100 for the simulation. This results in a timestep that is one fifth of that for an end time of 100 and we get the results shown in Fig. 4.7.

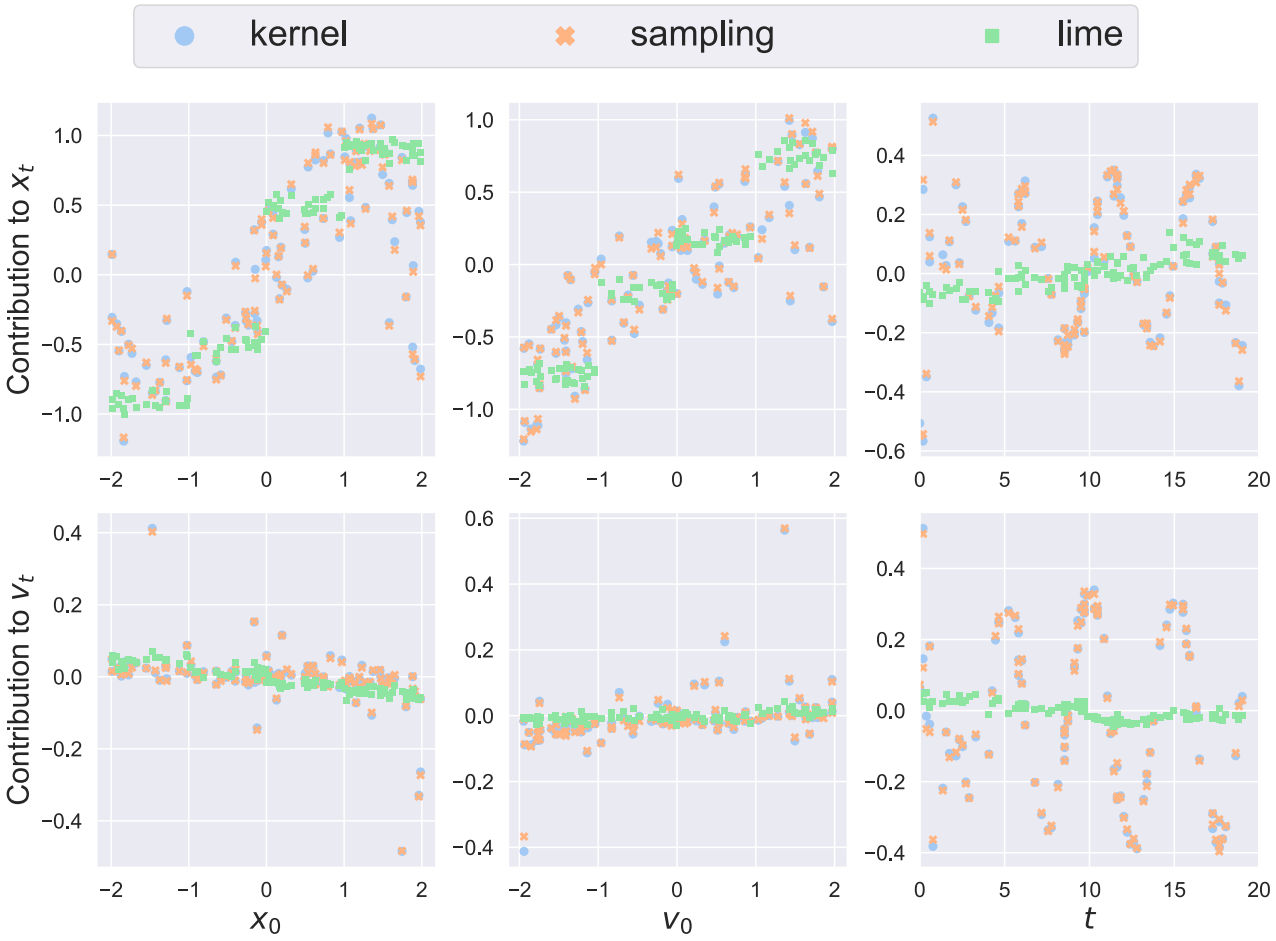


Fig. 4.7: Individual feature contributions in the damped double-well setting ($\alpha = -1$, $\beta = 1$, $\gamma = 0.37$, $\delta = 1.0$, $\omega = 1.2$). Simulation and evaluation performed for an end time of 20.

In this case we see that SHAP extracts the full oscillatory behaviour with the right period of oscillation ($T = \frac{2\pi}{\omega} \approx 5.2$). LIME does not assign the same level of importance to the time t for either of the two labels. This is likely because we have a kernel size that results in the drawn samples spanning multiple oscillations. When LIME tries to fit a linear model through these data it averages out the oscillations of the model that is being explained. For the feature importance attributed to x_0 and v_0 by LIME we see the values appear in 4 distinct blocks, this is the result of the quartile discretisation discussed in Sec. 3.1.1 but the overall trend is unchanged by this. Further, the values for t are unaffected by this setting.

We now look at the two-dimensional feature importance. Fig. 4.8. shows the kernel SHAP feature contributions as a heatmap over all values of x_0 and v_0.

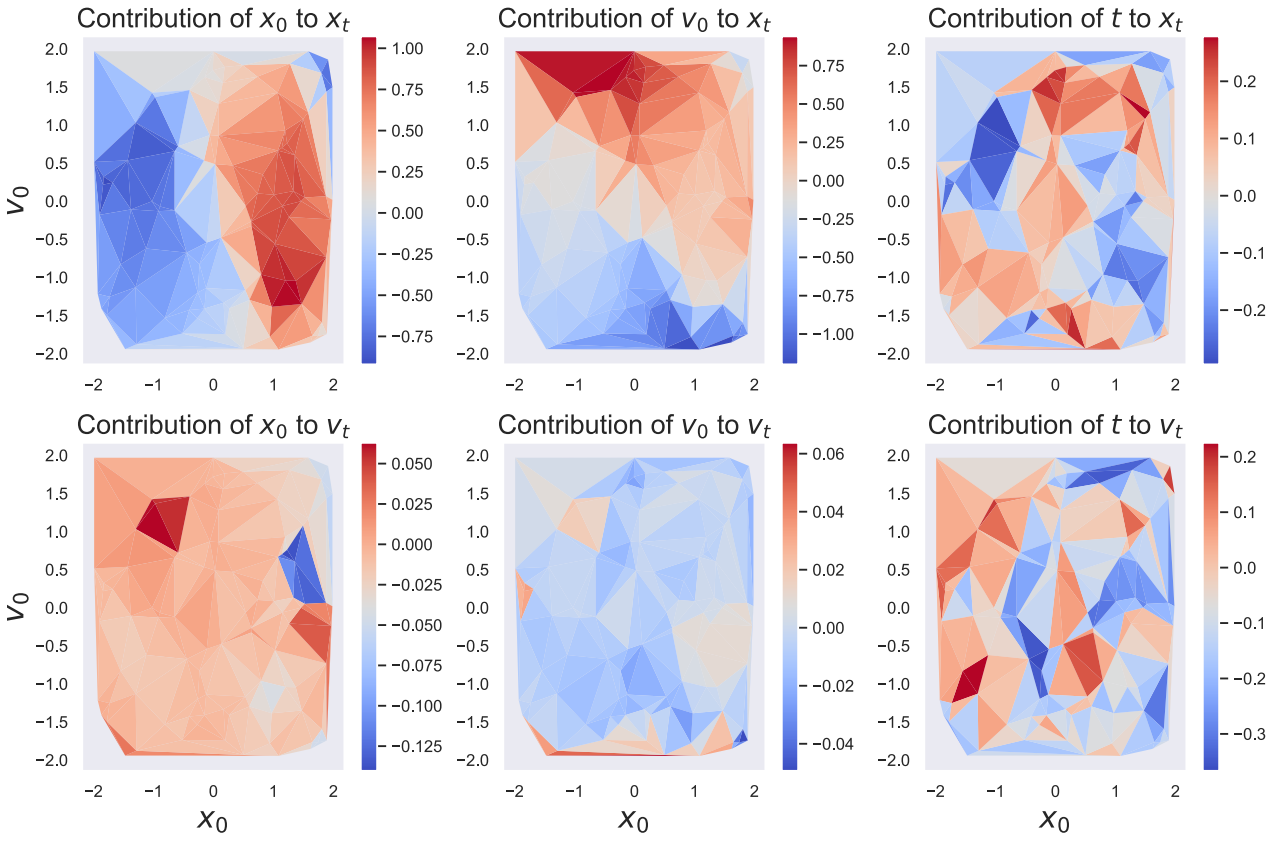


Fig. 4.8: Individual kernel SHAP feature attributions for the heavily damped double potential well system ($\alpha = -1$, $\beta = 1$, $\gamma = 0.37$, $\delta = 1.0$, $\omega = 1.2$) with the simulation. The values are plotted as a heatmap. The colour of each point denotes the value of the contribution to the respective label at the point (x_0, v_0)

The heatmap reveals the manner in which x_0 and v_0 interact when determining which well a point ends up in. The attributions are not exclusively dependent on x_0. The importances of x_0 and v_0 are correlated and v_0 can lessen the impact of the initial position x_0. When we have extreme values of v_0 the contribution of x_0 decreases, and the greatest negative feature attribution made to x_0 comes when $v_0 = 0$. Interestingly, the greatest positive feature attribution is recorded for $v_0 = -1.5$. The reasoning is that points located right in the potential well cannot be moved out of the well for values of v_0 around -1.5. Where a v_0 of -1.5 usually results in the trajectory ending up in the negative well, this is no longer the case here, reducing the importance of v_0 at this point and giving more weight to x_0. Looking at contributions made by v_0 paints a similar picture, but the extreme values of the feature attribution now follow the expectation of being found at $x_0 = 0$. The feature attribution to v_0 at $(1.0, -2.0)$ is around -1.0, this suggests that this is a velocity that can cause a transition to the other well, meaning that the velocity is important again. Looking back at the plot for feature contributions by x_0 to x_t (Fig. 4.8.) we see a confirmation of the theory we had as to why feature contributions dropped off so heavily for large values of x_0. We have extreme positive values of both x_0 and v_0 and yet the trajectory ends in the negative well. The high initial energy of the system allowing the transition to the negative well. The other plots do not yield any meaningful trends in the feature attributions.

Deep Neural Network

For the heavily damped double well, it is worth looking at how imperfect models impact the explainability. First, the Deep Neural Network (DNN) is examined. Fig. 4.9. shows the individual feature importance,

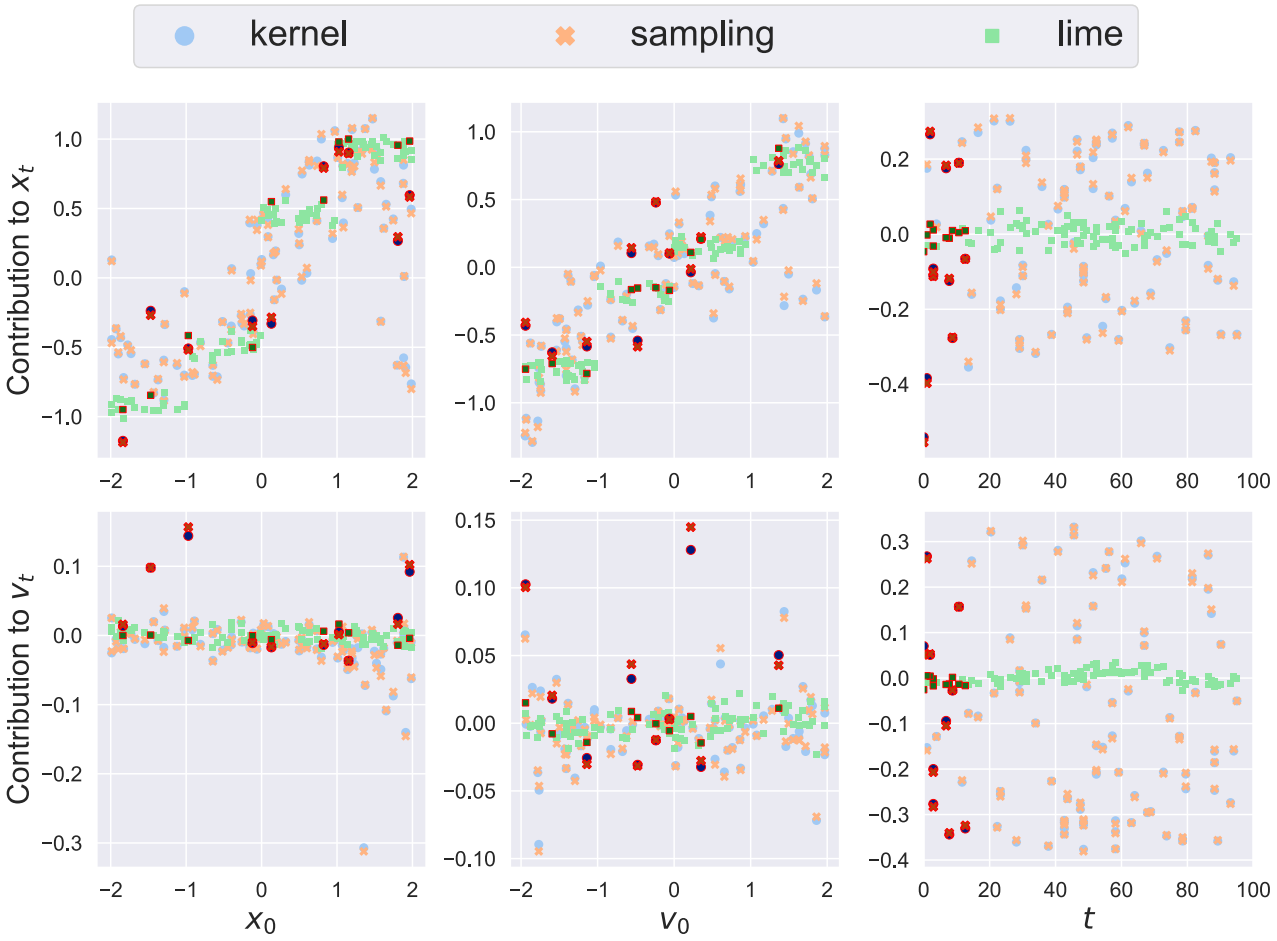


Fig. 4.9: Individual feature attributions for the heavily damped double potential well system with the deep neural network. Shown in darker colours are the worst predictions made by the model.

this is almost identical to the individual feature importance attributed to the simulation. This makes sense as the distribution of predictions made by the DNN is very similar to that of the predictions made by the simulation. The worst predictions made by the model have been plotted in a darker tone. These values are unified by the fact that they occur for low values of t. The fact that they disproportionately show a higher feature attribution to features that usually have low importance could suggest that the model is overemphasising this feature in this area. A more plausible explanation is that, since we are looking at values for low times, the initial position and velocity play a greater role in the predictions than usual. As mentioned before, for low values of t points are still able to transit the two wells, meaning that the system is more difficult to predict at this time. From these feature importances, we conclude that the model has learnt the basic nature of the system. It assigns points to wells based on their initial positions and velocities. It has also learnt the oscillatory behaviour we saw in the simulation. SHAP and LIME give us a way to visualise the causes for faulty model predictions. This insight into the failings of the model at low values of t could also be achieved by plotting the prediction error of the model against each feature. From this we could also identify that the model predictions are poor for low values of t.

Shallow Neural Network

The shallow neural network (SNN) was significantly worse at modelling the data than the DNN. As such, what we want from SHAP and LIME is some indication that this model isn't functioning as it should. This would make the explainers useful in determining when a model is insufficient to model the data.

In Fig. 4.10. we plot the individual feature importance for the SNN.

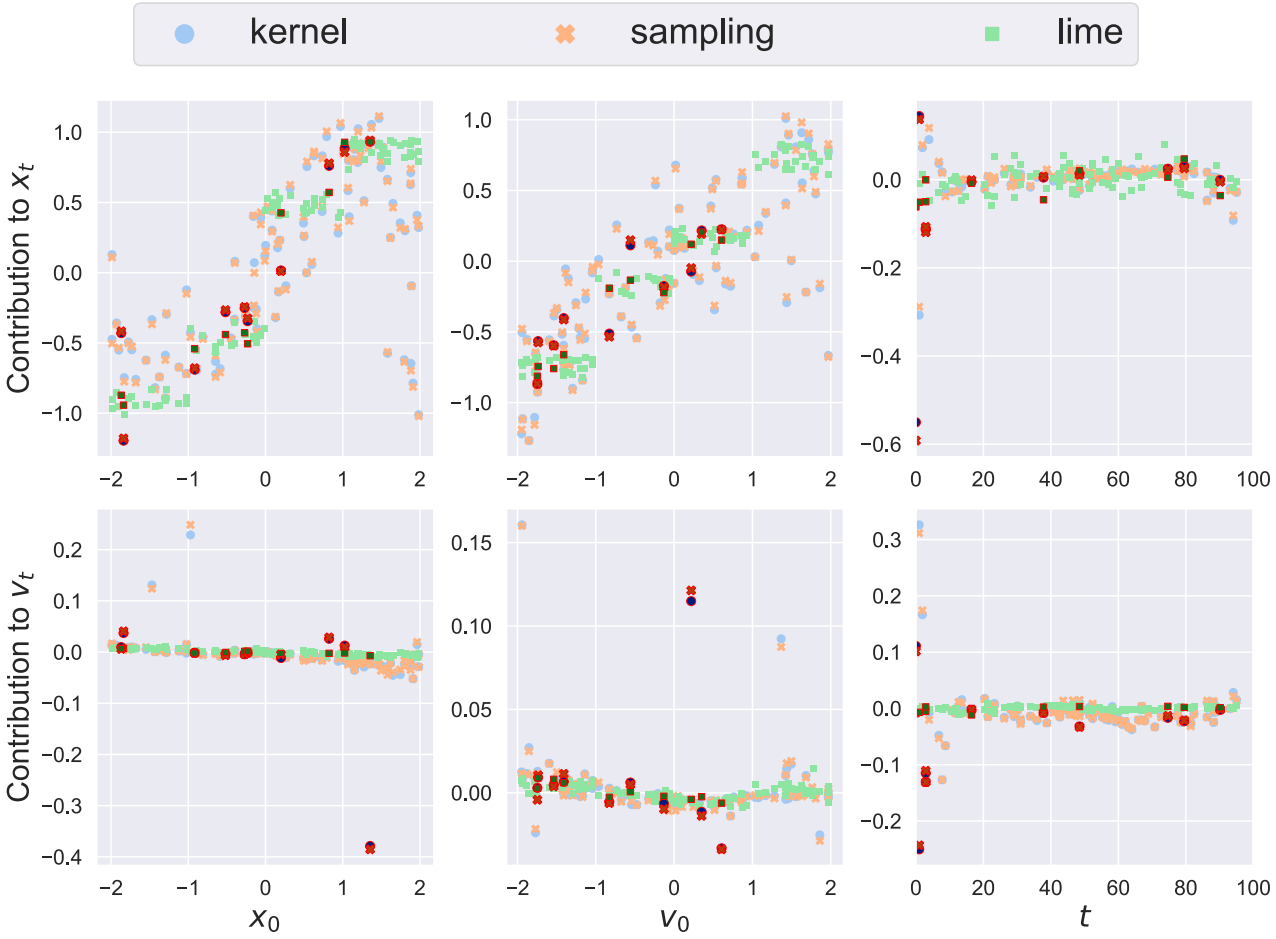


Fig. 4.10: Individual feature attributions for the heavily damped double potential well system with the SNN. Shown in darker colours are the worst predictions made by the SNN.

From this figure, we see that LIME has the same results for the SNN as for the DNN. This disqualifies it from being able to discern a model that has not learnt the system to a sufficient degree. SHAP, on the other hand, differs from the previous systems in that the time is now significantly less important. This makes sense as the SNN drags points to the centre of the potential wells and then doesn't feature any oscillatory behaviour (see Sec. 3.2.2.). SHAP correctly attributes low feature importance to t. Given a system for which we expect a contribution of t, looking at this feature importance does tell us where the model is going wrong. The SNN never learns the fact that points in the system oscillate around the potential wells. Looking at the poor predictions, these are now no longer all at low values of t, suggesting the model never learns the behaviour of the system, even when that becomes simple.

Gradient Explainer

The heavily damped double potential well is a system for which we have an understanding of what is to be expected. As such it makes sense to apply the gradient explainer to it, to see how far this very simplistic approach at explainability manages to emulate the results of SHAP and LIME. The individual attributions from the gradient explainer are shown in Fig. 4.11.

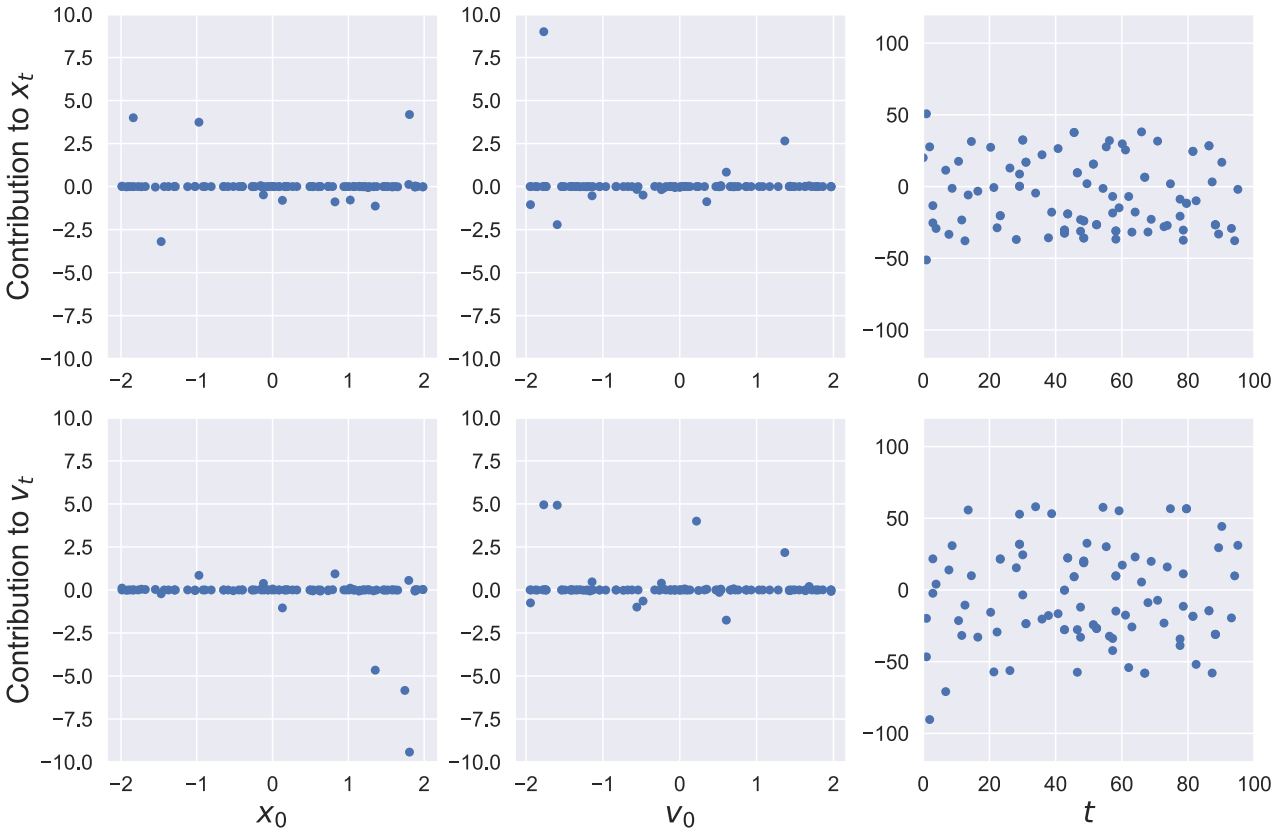


Fig. 4.11: Individual feature attributions, as determined with the gradient explainer, for the heavily damped double potential well system. The model being evaluated is the simulation.

where the figure is constructed in the same fashion as those for SHAP and LIME. From the figure, it becomes apparent that this explainer fails at extracting the behaviour that SHAP and LIME extracted about the importance of the initial position. Moreover, there are significant outliers in the feature attribution. It does manage to extract the behaviour that SHAP suggests for the importance of t. It attributes greater importance to low values of the time. For higher values of the time the spread of the feature importance then becomes constant. The gradient explainer is not delivering the results that we expected and also does not align with the other explainability techniques. This is likely because this approach is too simplistic for this oscillatory system. The gradient explainer is taking a similar approach to LIME in that it is fitting a linear model through data that it gets by perturbing the instance we are trying to explain. However, the gradient explainer is only using two data points, this seems to be insufficient to approximate this system locally. More research into this explainability method is orthogonal to the work we are trying to accomplish here.

4.1.3 Single Well

The final configuration we look at is the single well defined by the parameter configuration: $\alpha = 1.0$, $\beta = 1.0$, $\gamma = 0.37$, $\delta = 0.3$, $\omega = 1.2$. For this configuration all points end up in the same well, oscillating synchronously. Since all points end up at the same position regardless of (x_0, v_0) we expect SHAP and LIME to extract that these do not contribute. The individual feature contributions are shown in Fig. 4.12.

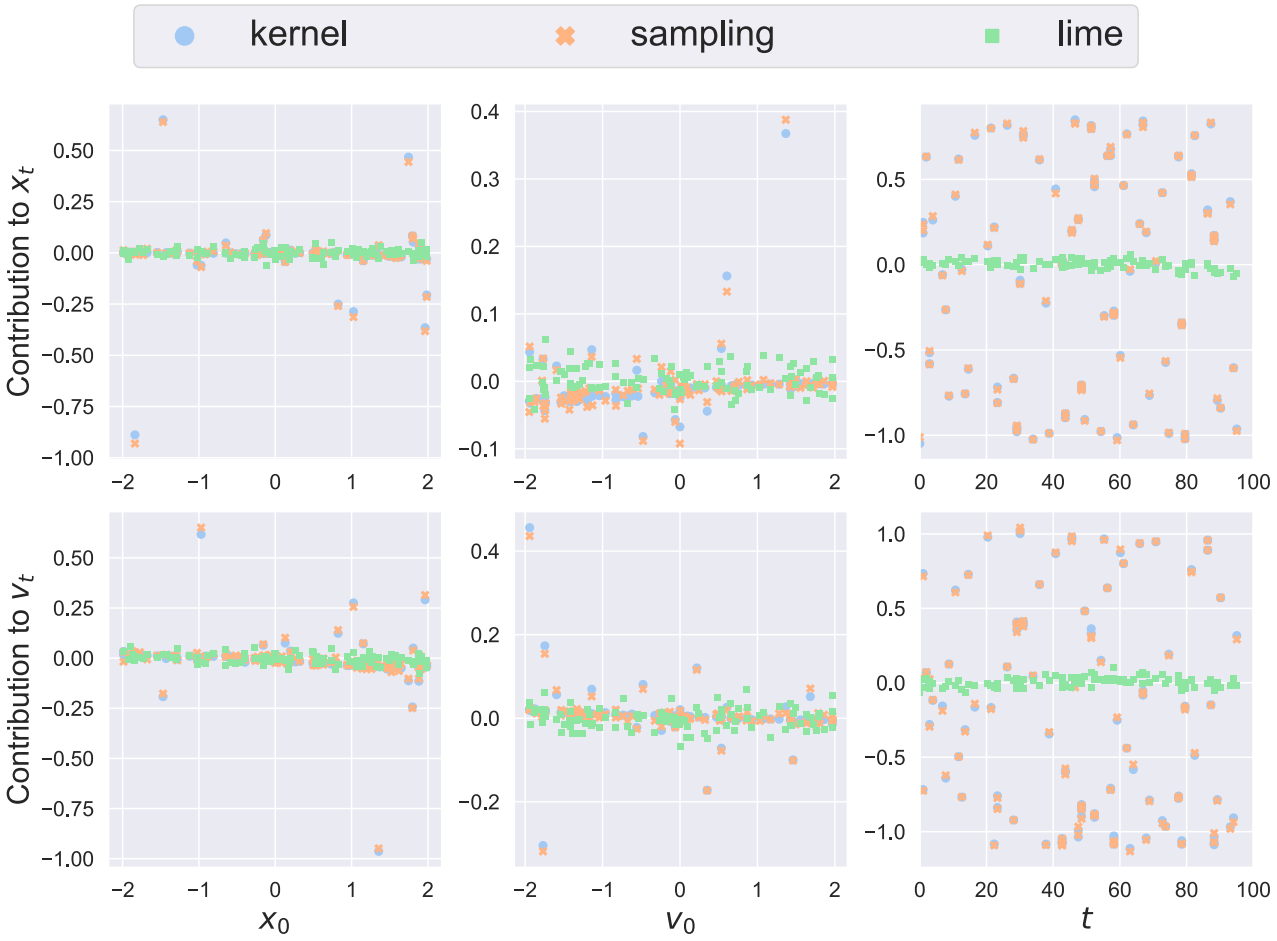


Fig. 4.12: Individual feature attributions for the single potential well ($\alpha = 1.0$, $\beta = 1.0$, $\gamma = 0.37$, $\delta = 0.3$, $\omega = 1.2$). Plotted are the feature contributions against the corresponding feature for each of the three explainer types.

This shows the expected behaviour. The features x_0 and v_0 do not contribute according to both SHAP and LIME. LIME also claims that the time t is irrelevant as before (Sec. 4.1.2.). SHAP shows the same oscillatory behaviour for the contribution of t that we saw with the damped double potential well.

4.1.4 Section Summary

When looking only at the relevant input features, SHAP functioned as expected. It was able to extract the feature contributions we predicted for the systems whilst being unable to extract intuitive feature importance for the lightly damped system. LIME displayed an inability to extract the importance of the time t for any system. This is likely due to it averaging out oscillations in its training of the linear surrogate models. However, this behaviour was consistent through all kernel widths tried in Sec. 3.1.1. Likely we would have to tune LIME to enable very low kernel widths for the time to get feature attributions for it. SHAP was able to extract the oscillatory nature of the system when applied to a system with a higher temporal resolution, but experienced aliasing for the system with an end time of 100. The gradient explainer is unable to extract anything about the importance of the initial position x_0 and initial velocity v_0.

4.2 Adding an Irrelevant Training Feature

The next thing to look at is whether the explainers are able to extract the fact that a feature isn't contributing to the outcome at all. This is an interesting setting as success here could help eliminate features that aren't needed to fit the model. If this can be done then these values wouldn't have to be measured in order to train the network, saving the effort required to gather this data. Removing irrelevant features also results in reducing the model complexity and the computational cost of training it. We would expect this to work for SHAP with the simulation, as changing the random feature will not cause a change in the model output. For the neural

networks, this is the first look at seeing whether we can identify a feature that can be discarded from the training data. The setting here can be summarised by:

$$\begin{aligned} f &: \mathbb{R}^3 \times \mathbb{R} \to \mathbb{R}^2, \\ f(x_0, v_0, t, r) &\to (x_t, v_t). \end{aligned} \tag{4.2}$$

Where r is randomly sampled from [0,1]. From the previous case, we learned that the lightly and heavily damped double potential well was the most interesting system to look at from an explainability standpoint. As such, we will be looking at these systems only.

SHAP and LIME

The first look will be at the heavily damped double-well characterised by the parameter configuration: $\alpha = -1$, $\beta = 1$, $\gamma = 0.37$, $\delta = 1.0$, $\omega = 1.2$. This configuration delivered the most interpretable results last time. Shown in Fig. 4.13 is the individual feature importance in this case for the simulation. The random feature is not involved in the predictions made with this model and so we would expect it to have zero feature importance.

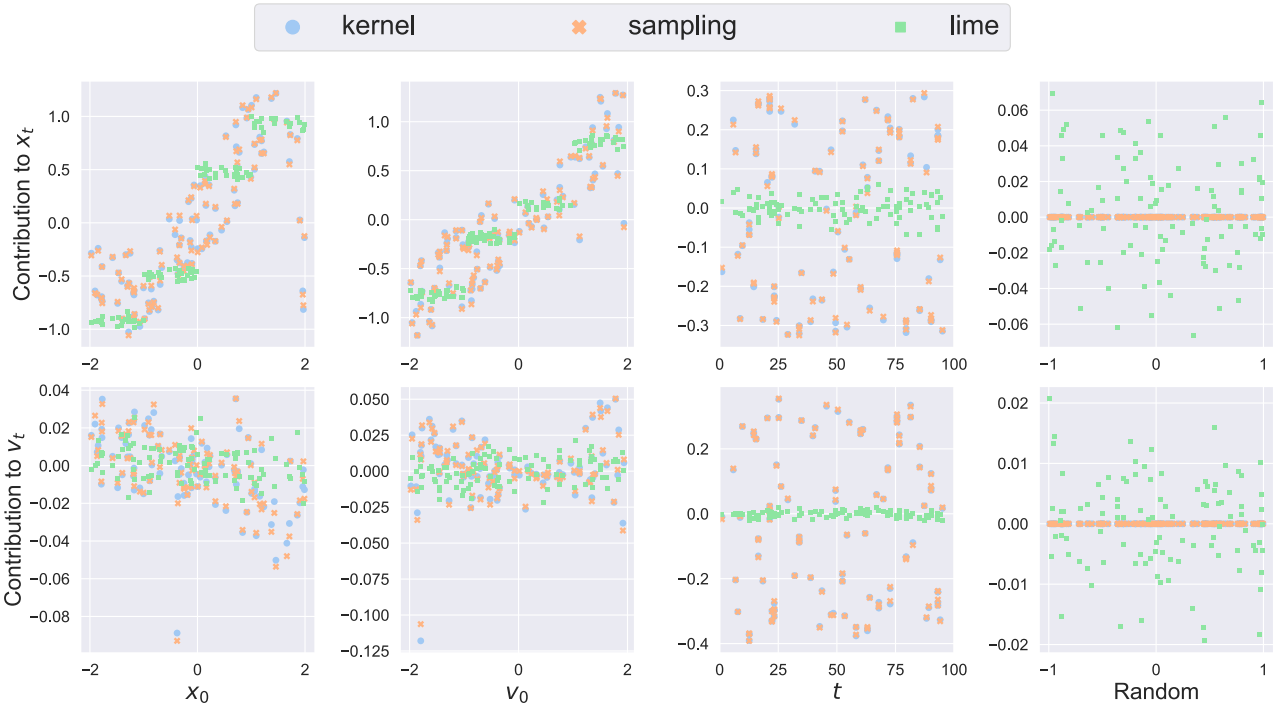


Fig. 4.13: Individual feature attributions for the heavily damped double potential well ($\alpha = -1, \beta = 1, \gamma = 0.37, \delta = 1.0, \omega = 1.2$) system with a random feature in the simulation.

The individual feature contributions are nigh identical to those for a system using only relevant features (see Fig. 4.5.). The added random feature has no impact on these values. This is to be expected as the simulated data points are not affected by the random feature. SHAP attributes zero feature importance to this feature whereas LIME gives it a very small contribution. Both explainers have noticed that the random feature doesn't contribute. SHAP has done so by noticing that passing different values of the random value through the model doesn't change the outcome at all. Where SHAP was altering only the random value LIME was perturbing all features simultaneously when training its linear model, this led to the LIME explainer finding non-zero importance for the random feature. Notably, this puts the scale of the contribution by the random feature to v_t on the same level as that of x_0 and v_0.

So SHAP is better at extracting that a feature doesn't contribute to a model at all. When we apply it to an imperfect model, such as the DNN, where the random feature may be taken into account this results in Fig. 4.14., where we plot the individual feature attributions for the DNN.

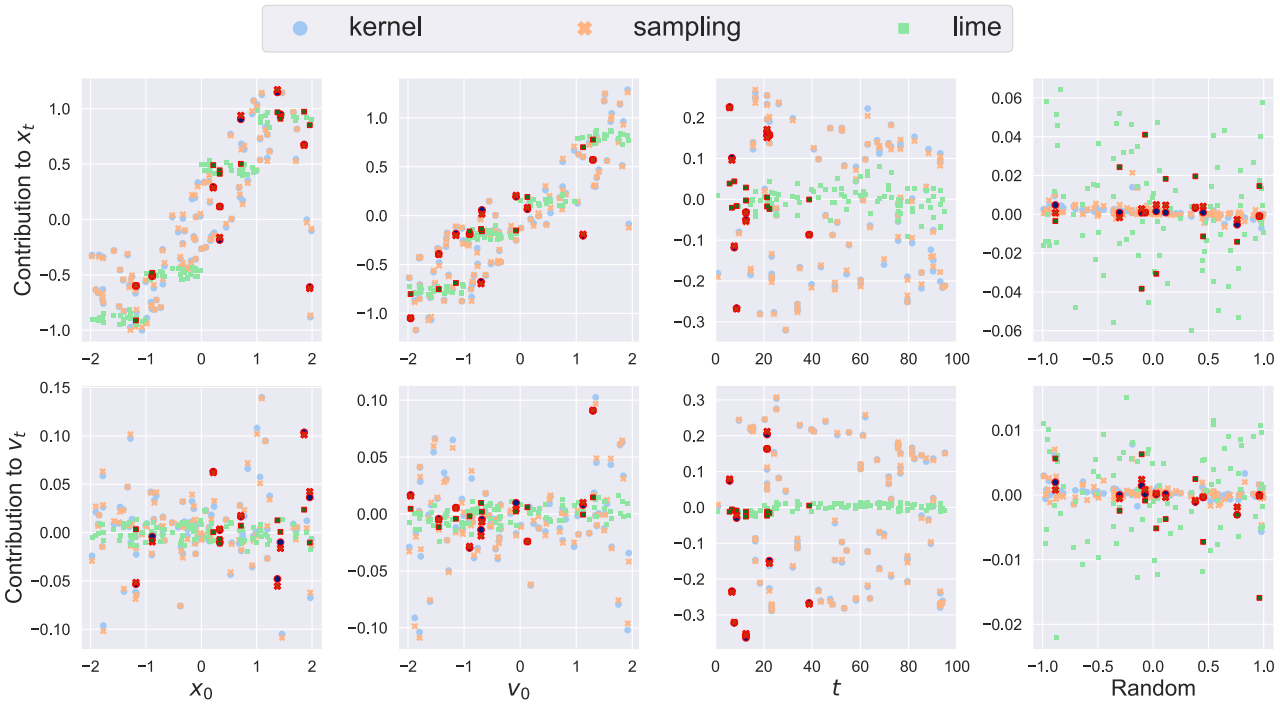


Fig. 4.14: Individual feature attributions for the heavily damped double potential well system with a random feature in the DNN. In darker colours are shown the worst predictions of the DNN.

From this, we see that the contribution of the random feature is no longer exactly zero, but is still very small. The explainer has concluded that the model does not incorporate the random feature in its predictions.

If the shallow neural network (SNN) were to suggest greater participation of the random feature then this could be a way to reason where the model is going wrong. However, as we can see in Fig. 4.15. the SNN also attributes very little importance to the random feature. Suggesting that it has also noticed that this feature is of no importance.

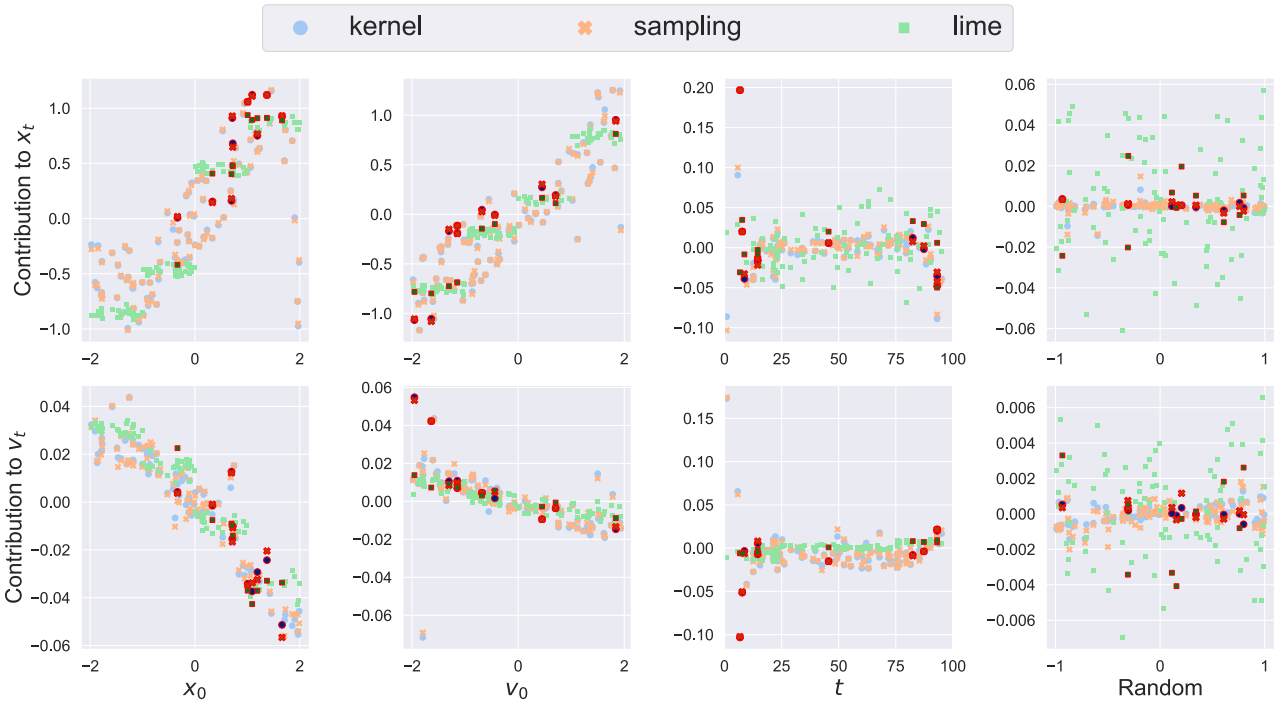


Fig. 4.15: Individual feature attributions for the heavily damped double potential well system with a random feature in the SNN. In darker colours are shown the worst predictions of the SNN.

But this is a system where the networks were able to reach a fairly good accuracy. What happens when we

apply them to the less predictable system of the double-well with light damping is shown in Fig. 4.16.

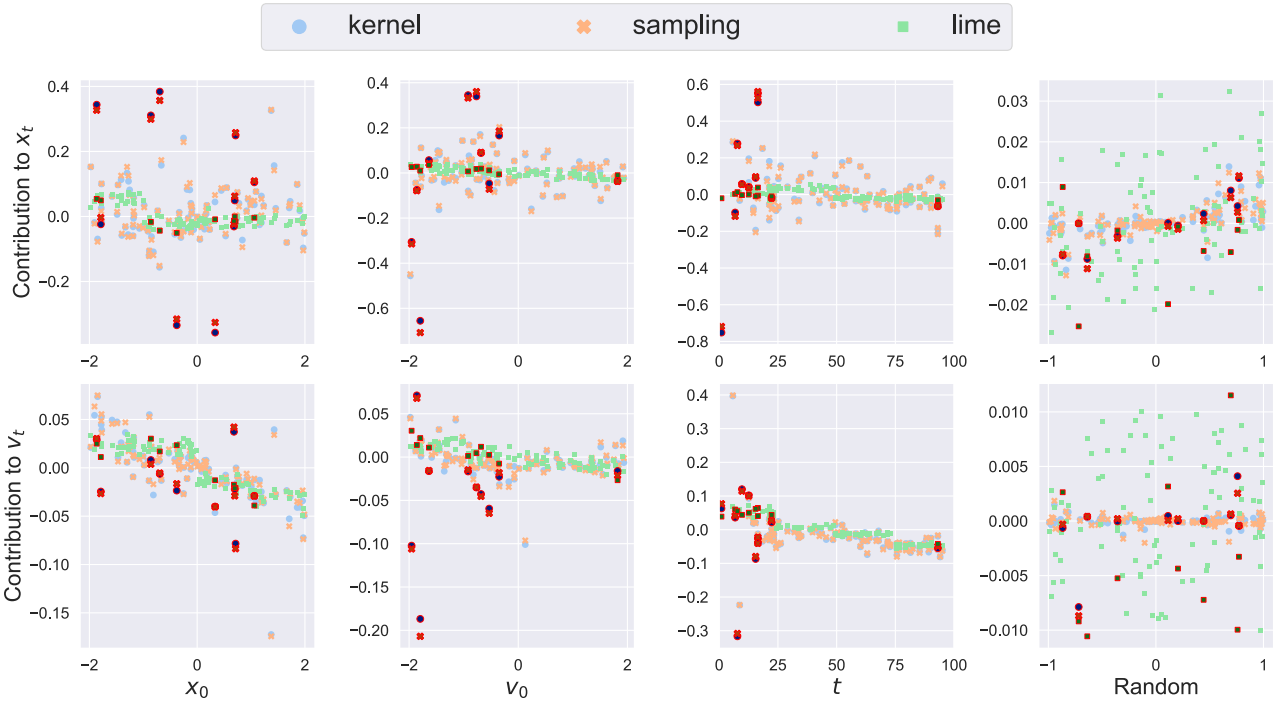


Fig. 4.16: Individual feature attributions for the lightly damped double potential well system with a random feature in the SNN. In darker colours are shown the worst predictions of the SNN.

Even in this case, the SNN has extracted that the random feature is not contributing to the outcomes of the simulation. Both the explainers and the models have succeeded in learning that the random feature does not contribute. This is encouraging, as this means that the explainers can indeed be used to identify and remove features that do not improve the model accuracy.

4.2.1 XGBoost

As mentioned in Sec. 2.1., these are not the only ways to extract feature importance from a model. XGBoost [5] is a take on the gradient boosting methods, that uses Newton Raphson instead of gradient descent to train decision trees. This method comes with the benefit that we can attribute a sort of feature importance by looking at the gain attributed to each feature. At each leaf in the tree a decision is made based on the value of a single feature. Each leaf in the tree splits into two and this decision decides on whether or not a left or right path is chosen in the tree. For each leaf we can define the gain as (Eq. 4.3.):

$$\text{Gain} = \frac{1}{2}\left[\frac{G_L^2}{H_L+\lambda} + \frac{G_R^2}{H_R+\lambda} - \frac{(G_L+G_R)^2}{H_L+H_R+\lambda}\right] - \gamma \tag{4.3}$$

, where $\frac{G_L^2}{H_L+\lambda}$ is the score on the new left leaf and $\frac{G_R^2}{H_R+\lambda}$ the score on the new right leaf. $\frac{(G_L+G_R)^2}{H_L+H_R+\lambda}$ is the score on the original leaf. The variable γ is introduced to punish unnecessary leaves and acts as a regularisation parameter for the complexity of the tree. Summing these gains for each leaf in which feature j is used in the tree gives us the F-score for feature j (Eq. 4.4.):

$$\text{F-score}_j = \sum_{\text{leaf}_j \in \text{tree}} \text{Gain} \tag{4.4}$$

This gives us something akin to a feature importance. We can compare this F-score "feature importance" to both SHAP and LIME in addition to comparing them to our expectations. Plotting the F-score for a model of the heavily damped double well gives us the results from Fig. 4.17.

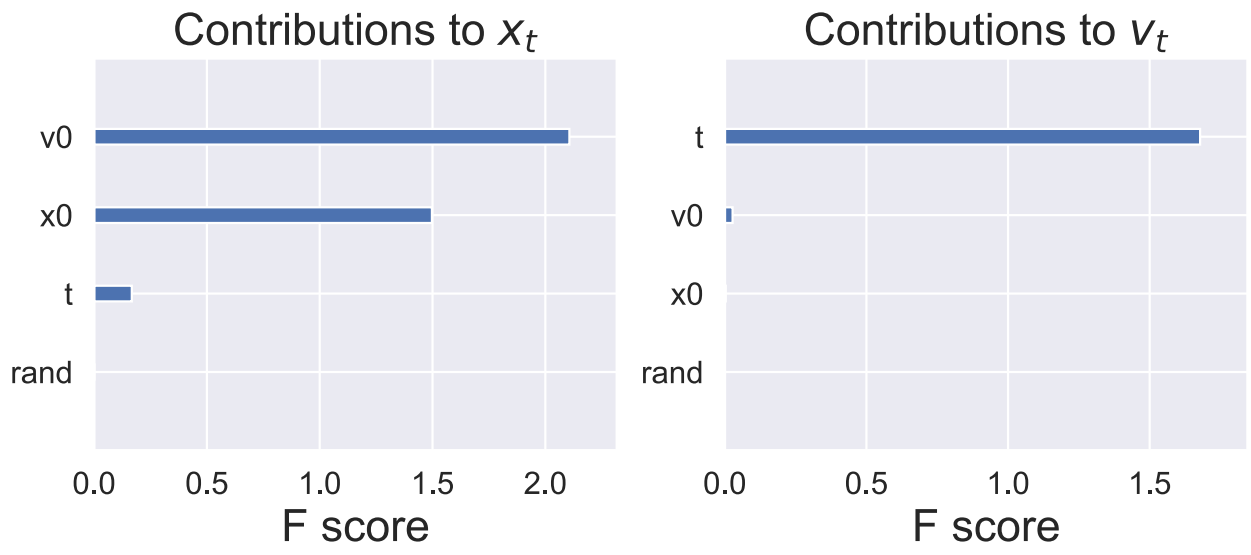


Fig. 4.17: Feature importance (gain) plot of an XGBoost decision tree for the heavily damped double-well with an irrelevant training feature. This plot used 150 trees with a maximal depth of 10. Left: Feature Attributions to x_t. Right: Feature Attribution to v_t.

This XGBoost model was generated with a maximum tree depth of 10 and cut off after around 150 trees. XGBoost claims that it used x_0 and v_0 predominantly when predicting $x(t)$. The time t also contributed in the prediction but the random feature was completely absent. For v_0 we see that t is the only important feature. This aligns very well with the aggregate feature importance assigned by SHAP (Fig. 4.18.),

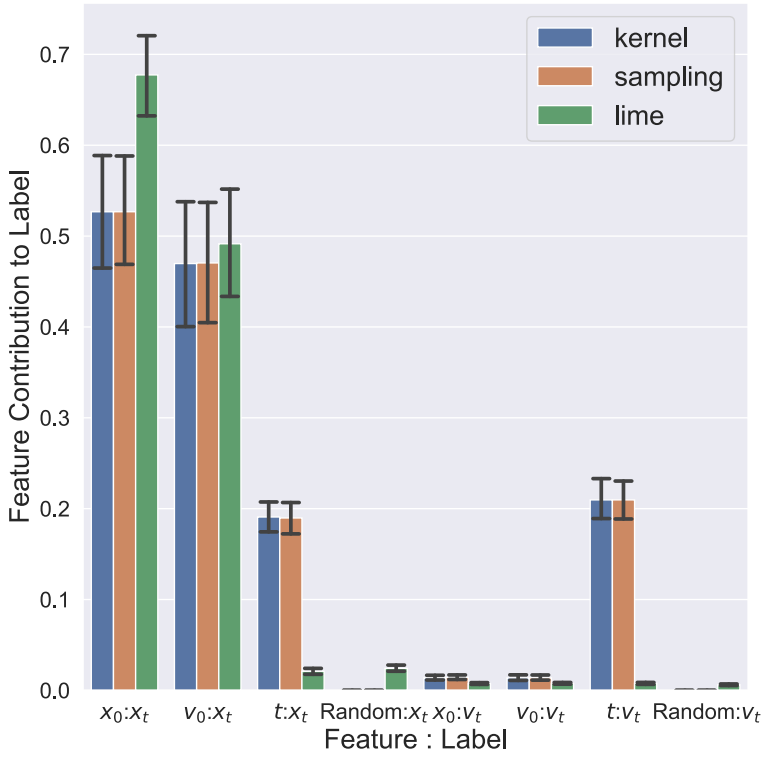


Fig. 4.18: Aggregate feature importance for the simulation in the heavily damped double-well configuration with an irrelevant feature as a training feature.

where we see the same behaviour for SHAP as we saw for XGBoost. Though LIME also exhibits the same behaviour for x_t it fails when we get to v_t and also over includes the random feature.

4.2.2 Section Summary

The inclusion of the random feature did not impact the feature importance for the relevant features for SHAP and LIME when looking at the simulation and the DNN. This is the expected result, as the simulation and the models still functioned as before. LIME did incorrectly attribute non-zero importance to the random feature. Comparing the results to XGBoost showed that SHAP and XGBoost's feature importance based on the "gain" agreed. We now look at a setup wherein the feature that isn't included in the simulation is a function of the relevant features x_0 and v_0.

4.3 $g(x, v)$ as a Training Feature

The function of x_0 and v_0 that we will look at is the initial energy of the system. This results in having to model:

$$\begin{aligned} f &: \mathbb{R}^3 \times \mathbb{R} \to \mathbb{R}^2, \\ f(x_0, v_0, t, g(x_0, v_0)) &\to (x_t, v_t). \end{aligned} \tag{4.5}$$

We define the energy as ([16]):

$$g(x, v) = \frac{1}{2}v^2 + \frac{1}{2}\alpha x^2 + \frac{1}{4}\beta x^4. \tag{4.6}$$

In this case, the energy is correlated with the initial position and velocity. It should be noted that changing the energy in our data generation does not change the initial position or velocity, but the converse does affect a change in the energy. Further, the feature contains information about the potential of the system. What we have learnt of SHAP suggests that SHAP should be oblivious to this new feature when applied to the simulation. The question is what happens when we apply it to a machine learning model that may be using the energy to make its predictions. It should discern that, whilst this feature is linked to the relevant features, it has no effect on the predictions of the model, and thus, no attribution should be made to it. Fig. 4.19. depicts the feature attributions made by the three explainers for the damped double-well system with the initial system energy as a training feature.

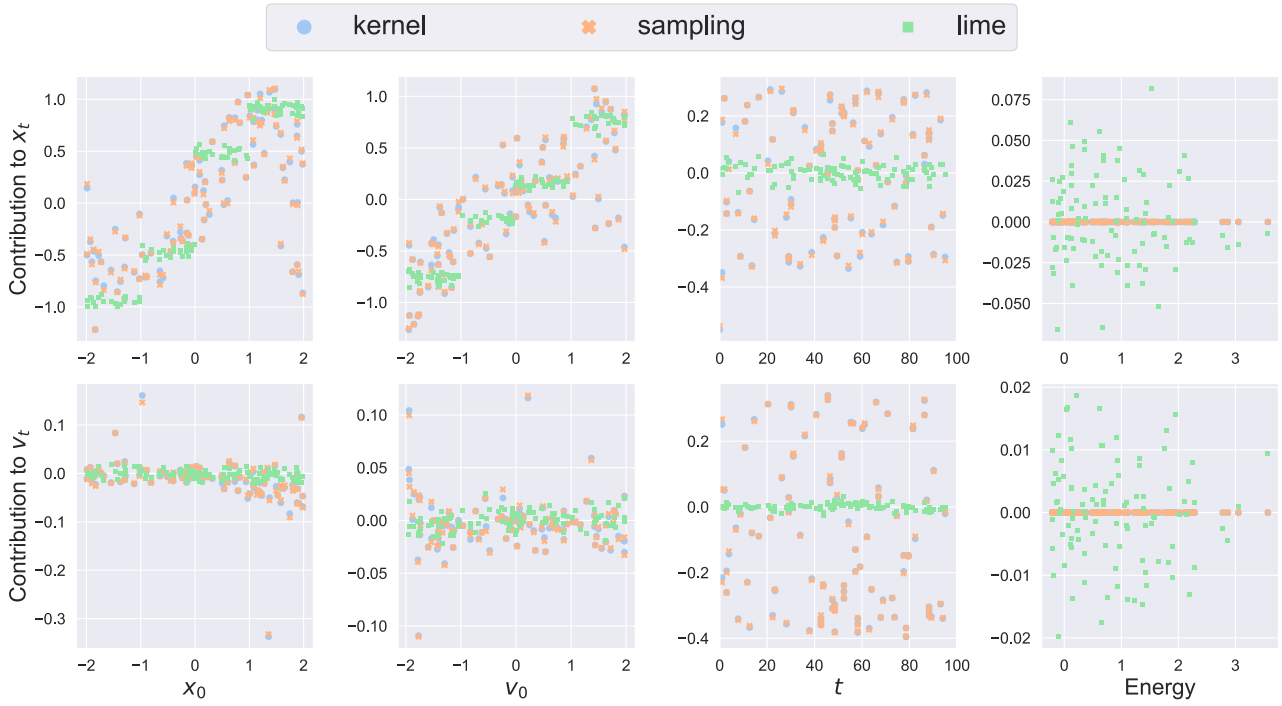


Fig. 4.19: Individual feature attributions to the simulation in the heavily double-well configuration. We use the initial system energy as a training feature.

From the figure, we see that, for the simulation, SHAP was able to replicate the success from the random feature. It has correctly identified that the energy is a correlated feature that does not contribute to the model output.

LIME attributes a feature importance to the energy that is on par with the time t. This is a result of LIME altering multiple features simultaneously when selecting samples to train the linear model. When it does this, different energies correspond to different initial conditions and thus affect the prediction in the same manner as for the random feature (Sec. 4.2.). The fact that this implementation of LIME does not test the model with custom samples, where only single features are perturbed, makes it difficult for LIME to determine when a feature is contributing or merely correlated to the relevant training features.

Evaluation on the DNN could yield a dependence on the energy if the model has learnt that this correlation exists and is using it to predict data points. The individual feature attributions for the DNN are shown in Fig. 4.20.

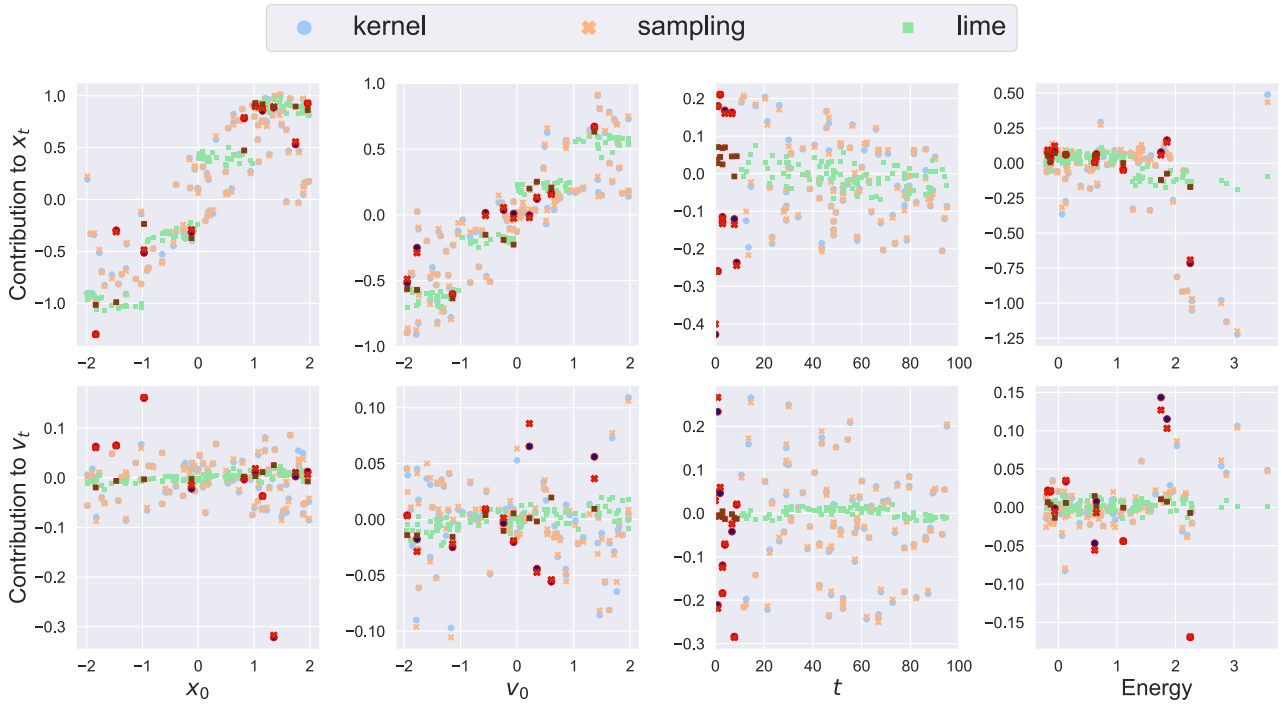


Fig. 4.20: Individual feature attributions to the DNN in the heavily damped double-well configuration. We use the initial system energy as a training feature. Shown in darker colours are the 10 worst predictions made by the DNN.

This figure shows that the model is using the energy for its predictions for higher energies. The contributions according to SHAP increase drastically for values of the initial system energy $E \approx 2$. There are very few points for these high values. These may be points for which the initial energy is enough to facilitate a transfer between the two potential wells. In this case, the DNN may be using the energy to decide on when to move a point to the other side of the potential hill between the wells. The reason for the points being predominantly negative in their contribution to x_t at higher energies is a result of the low sample size.

XGBoost

We can look at XGBoost again to see how having a feature that can be used to gain information about the system impacts the feature importance that XGBoost attributes. The predictions XGBoost makes could be contingent upon the energy as well as the "relevant features". The feature importance that XGBoost suggests for this system is shown in Fig. 4.21.

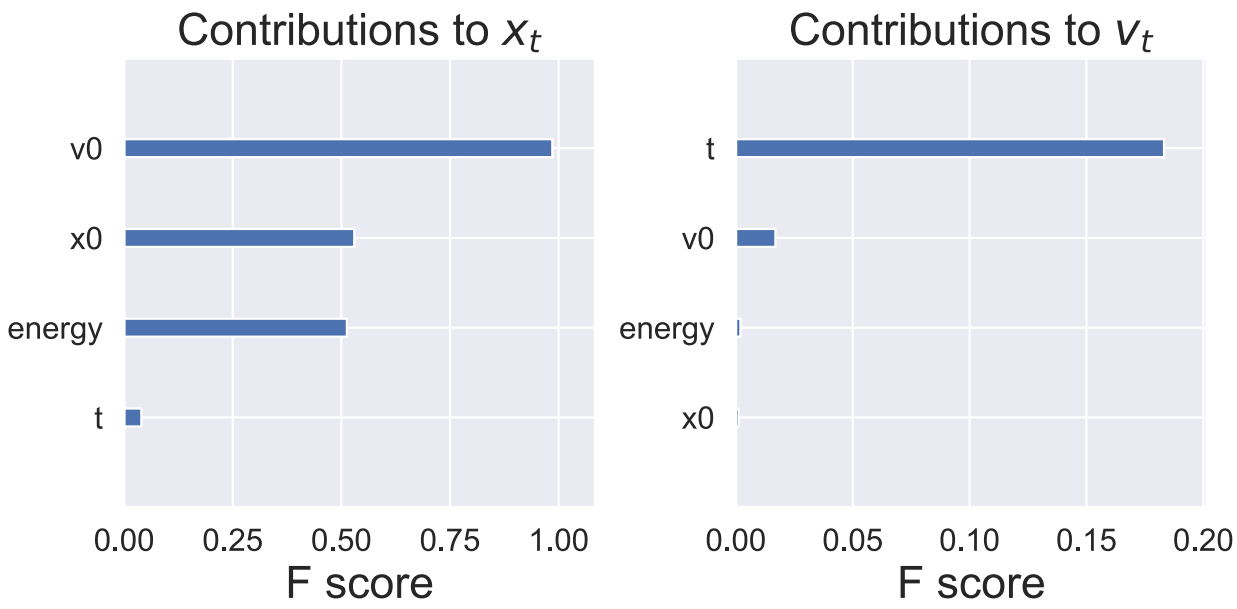


Fig. 4.21: Feature importance (gain) plot of an XGBoost decision tree for the heavily damped double-well with the initial system energy as a training feature. This plot used 150 trees with a maximal depth of 10. Left: Feature Attributions to x_t. Right: Feature Attribution to v_t.

The feature attributions that XGBoost provides are different from those provided by SHAP and LIME (see Fig. 4.22.).

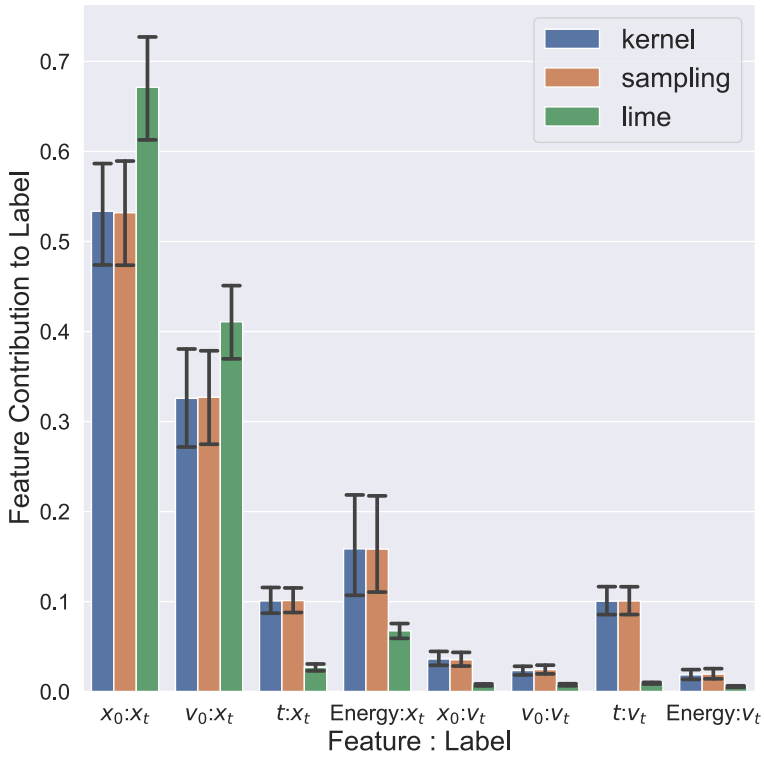


Fig. 4.22: Aggregated feature attributions for the lightly damped double potential well ($\alpha = -1$, $\beta = 1$, $\gamma = 0.37$, $\delta = 0.3$, $\omega = 1.2$). We include the initial system energy as a training feature. Explainability methods LIME, SHAP (kernel) and Shapley (sampling) values are applied to the DNN. Data aggregation is performed by taking the mean of the absolute values. Plotted in black are the 95% confidence intervals, determined by bootstrapping.

According to XGBoost, the energy is now as important as the initial position. This is likely because these

two features are highly correlated. Using v_0 and the energy, x_0 can be derived. When faced with duplicate features, decision trees may choose at random which feature to use in making the decision, we may be seeing this effect here.

4.3.1 Section Summary

SHAP did not attribute a meaningful contribution to the energy when looking at the simulation, as expected. The attributions suggested by LIME seem to stem from the amount of data available for a given energy as the feature attributions decrease in absolute value as we get to higher energies where data points are sparser. When looking at the DNN we see that the energy contributes for higher energies. As previously explained, this could suggest that the model is using this information to predict a well transit. XGBoost may be treating the initial system energy as analogous to the initial position x_0 in its predictions.

4.4 Adding Global Model Parameter γ as a Training Feature

Now we look at the attributions SHAP and LIME make to a global feature (γ). In this case, we look at the amplitude of the driving force in the Duffing Oscillator (Eq. 2.8.). This case is interesting as, if it is possible to extract the fact that γ is a global parameter from the explainer results, we could use this in other cases to find features that change the underlying physics of the problem itself. This setting is summarised by:

$$\begin{aligned} f &: \mathbb{R}^3 \times \mathbb{R} \to \mathbb{R}^2, \\ f(x_0, v_0, t, \gamma) &\to (x_t, v_t). \end{aligned} \tag{4.7}$$

4.4.1 Networks

Since γ changes the data generation process itself, the neural networks are greatly affected by this setting. As such, we examine the performance of these networks as we did for the basic case in Sec. 3.2. Once again, we look at the lightly and heavily damped double-well system. For each system, the data is generated so that γ is drawn from [0,1] with the step size between adjacent values of gamma being 0.01.

The model performance for the lightly damped system is shown in Fig. 4.23.

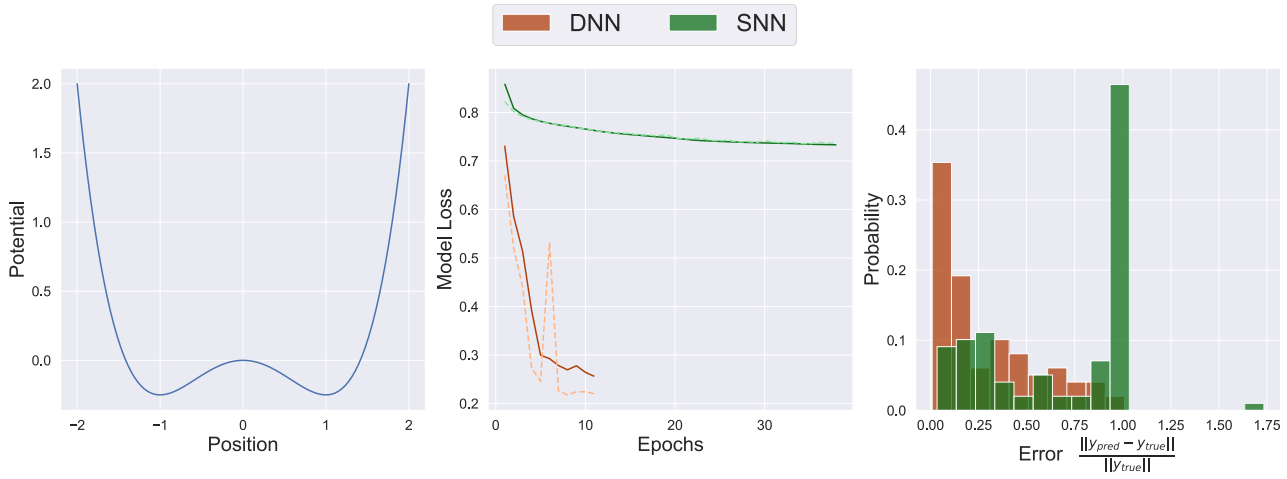


Fig. 4.23: Model Training summary for the lightly damped ($\delta = 0.3$) double-well ($\alpha = -1, \beta = 1$) potential with the driving amplitude γ as a training feature. The potential is shown on the left, in the centre are the model training (continuous) and validation (dashed) loss for the shallow (SNN) and deep (DNN) neural networks. On the right is an error plot that compares the error on the validation set for the two models, outliers, the largest 5% of the error, are removed.

The shallow neural network (SNN) performs poorly once we add γ as a feature, with the training and validation losses levelling out at a $MSE \approx 0.7$. The deep neural network (DNN) trains quickly, and could even have benefitted from more training. Looking at the model predictions in Fig. 4.24.,

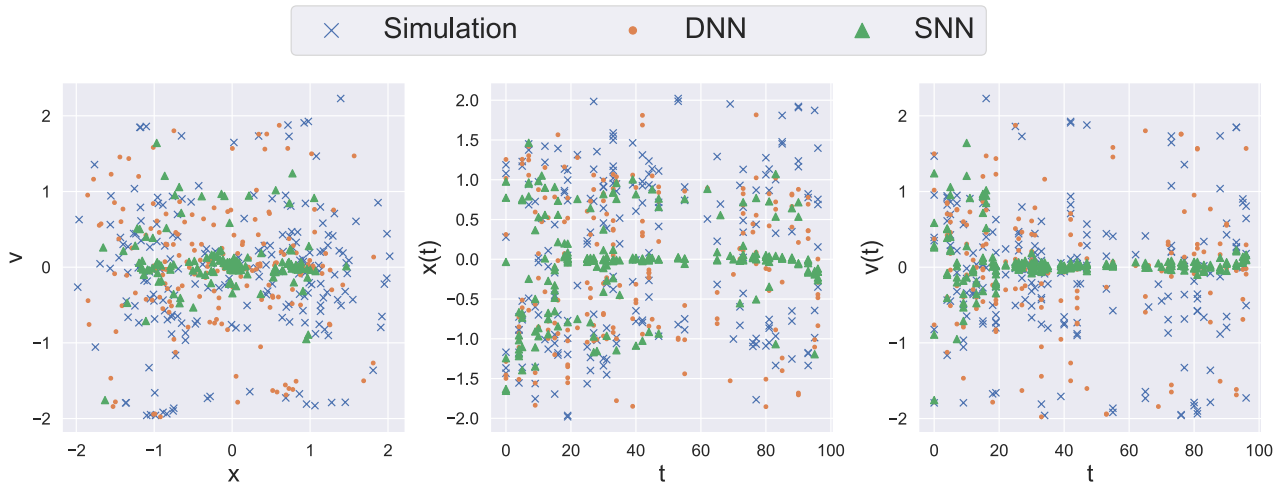


Fig. 4.24: Predictions made by the machine learning models for the weakly damped double-well potential. Left: Plot of the model outputs. Centre: Plot of $x(t)$ by t. Right: Plot of $v(t)$ by t.

shows us that the networks here follow the trend set out by those for the setting with only (x_0, v_0, t) as input features. The SNN is insufficient to model this more complex system. The DNN is able to model the system with roughly the same accuracy as the lightly damped double well despite the increased complexity when compared to that system. The model performance is worse than for the systems without γ as a training feature. The network training metrics for the heavily damped double-well and the single well are found in Appendix A.

4.4.2 Explainability

Lightly Damped Well

Now we apply the explainers to this setup. In Fig. 4.25. we plot the individual feature attributions for the lightly damped double-well.

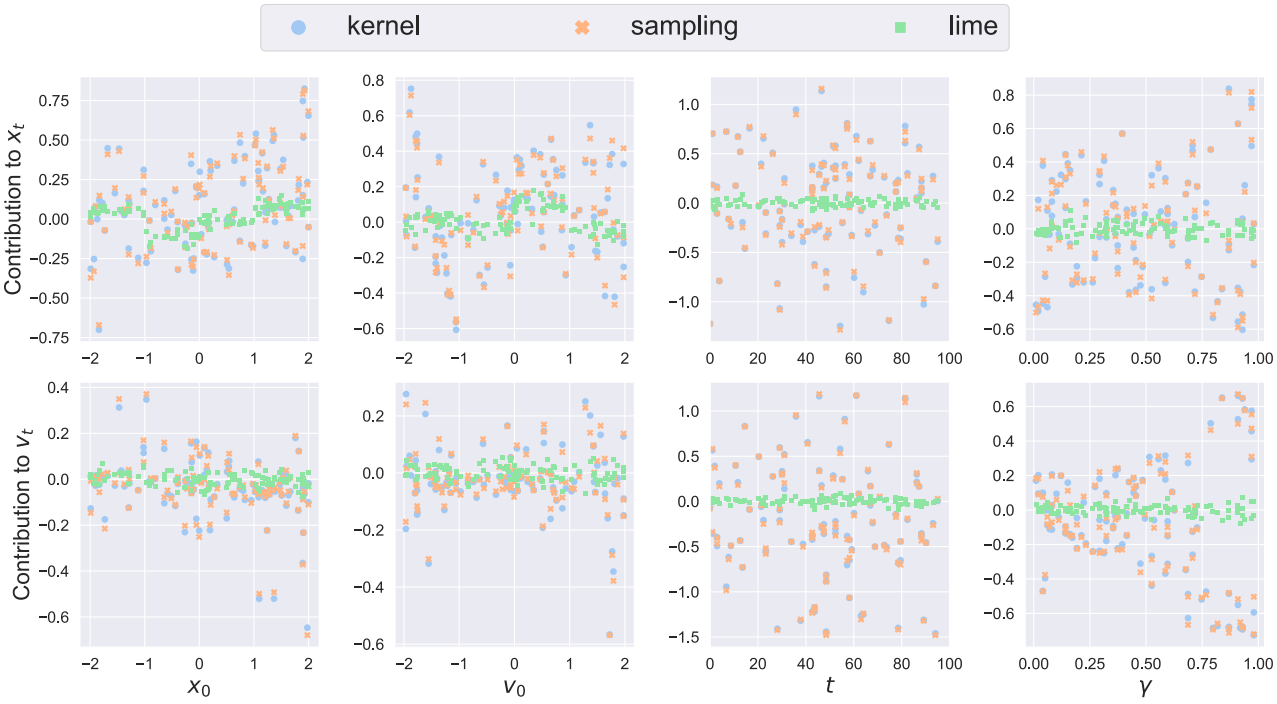


Fig. 4.25: Individual feature attributions for the lightly damped system with variable γ in the simulation. Plotted are the feature attributions against the corresponding features.

From this figure, we see that the model attributes feature importance on par with that for x_0 to γ. With the importance of γ increasing with γ. What we cannot say from this plot is whether γ is contributing in a

different fashion to x_0 or v_0. From this plot, we judge that LIME is unable to extract any information about the impact γ has on the system.

Looking at a plot of the feature contributions against γ (Fig. 4.26.),

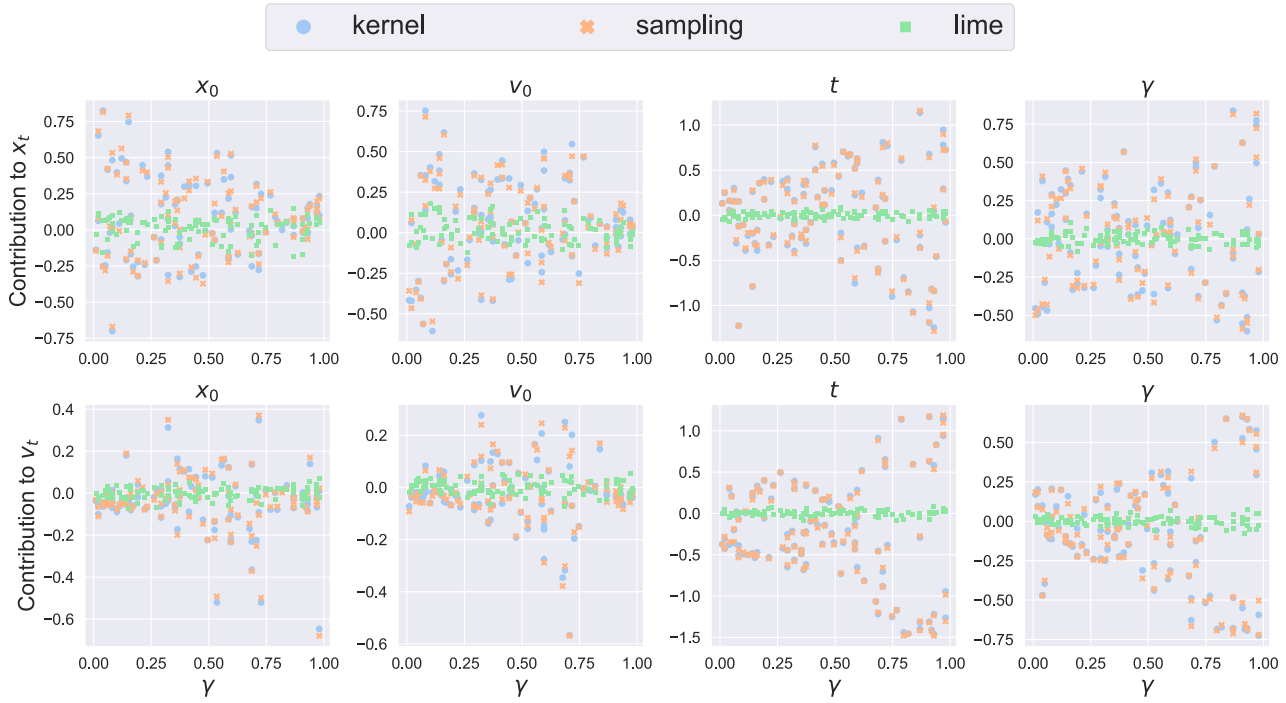


Fig. 4.26: Individual feature attributions for the lightly damped system with variable γ in the simulation. Plotted are the feature attributions (y-axis) against γ (x-axis).

we see that x_0 and v_0 fall off in their contribution to x_t for increasing γ, this is to be expected. As we increase the amplitude of the driving force, more points have the energy to switch the well they are in. As the amplitude reaches a certain point points can switch freely between the two wells and we get very low contributions by x_0 and v_0. For t the contributions grow with γ because the driving force governs the only t dependent term in the equation of motion ($\gamma \cos(\omega t)$).

Heavily Damped Double-Well

We now look at the heavily damped system, as we know more about the expected feature importance in this setting. What we are doing when we change γ is increasing the amplitude of the driving force. This means that for larger γ we compensate the damping more, allowing points to switch wells. When we have light damping this does not have an effect as points are already transiting wells. What we expect to see here is a drastic increase of the importance of γ as we reach a threshold, past which points are able to pass between the wells. Shown in Fig. 4.27 is the aggregate feature importance in this system.

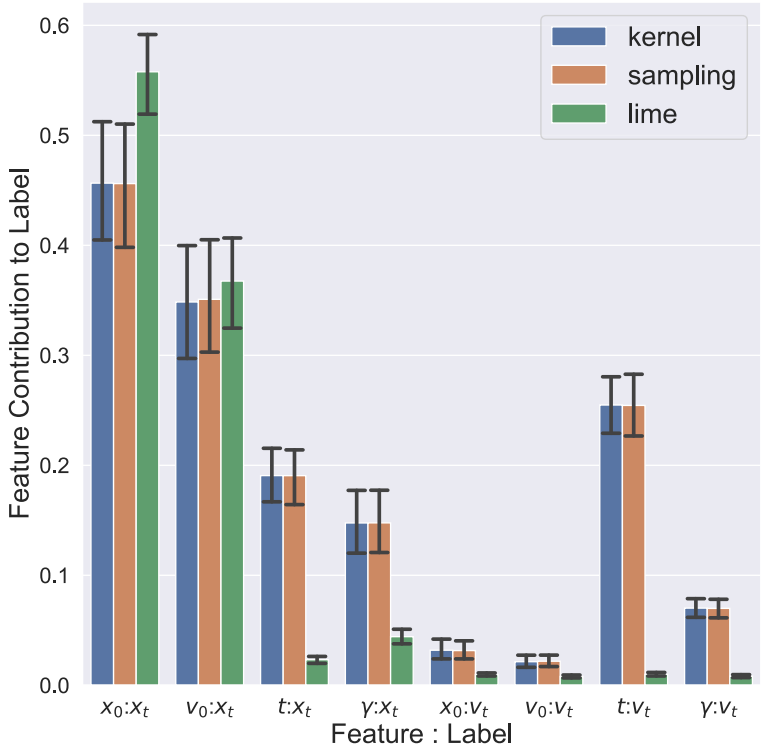


Fig. 4.27: Aggregate feature importance for the heavily damped system with variable γ for the simulation. Data aggregation is performed by taking the mean of the absolute values. Plotted in black are the 95% confidence intervals of the aggregated importance, determined by bootstrapping.

This plot suggests that γ is far less important in this setting than in the lightly damped case, this goes against what we expected. Using SHAP and LIME allows us to take a closer look at what's going on by looking at the individual feature importance in Fig. 4.28.

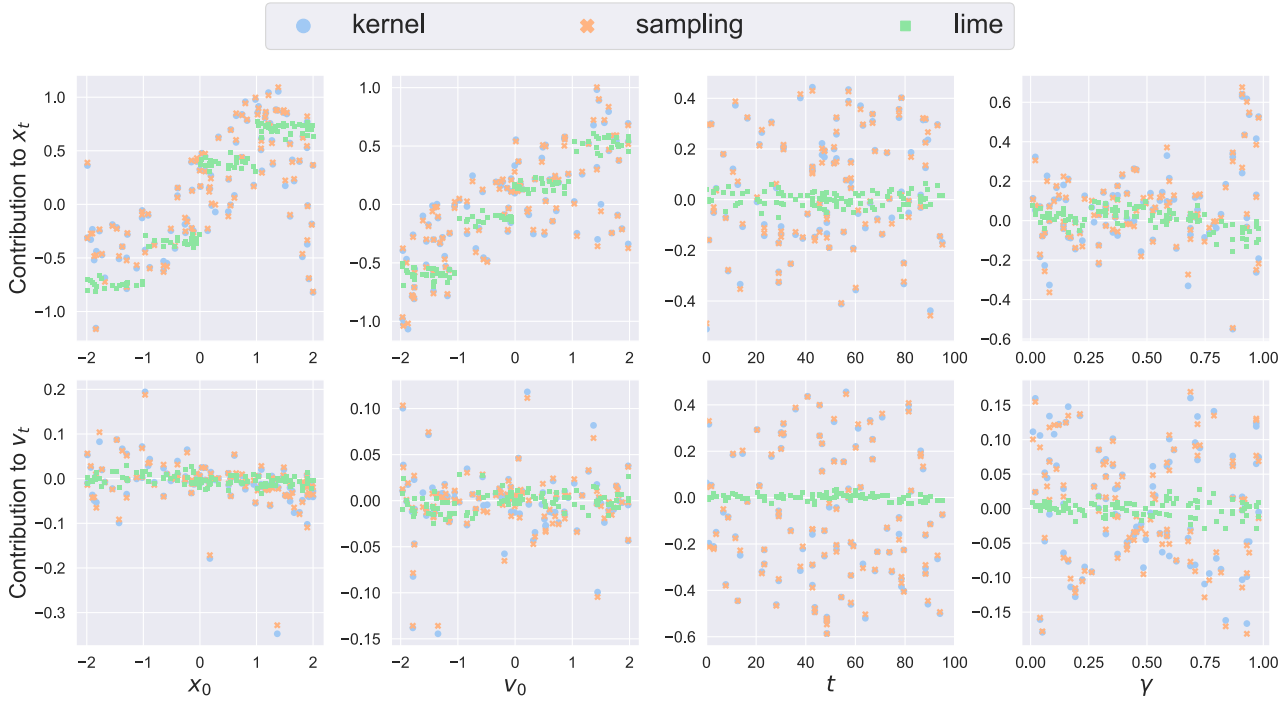


Fig. 4.28: Individual feature attributions for the heavily damped system with variable γ in the simulation. Plotted are the feature attributions (y-axis) against the corresponding features (x-axis).

From this plot, we now see the expected behaviour for large γ. Whilst γ does contribute very little at low

values of γ, its importance increases dramatically as we get to $\gamma \approx 0.8$. This suggests that this is a value of γ for which the system has undergone a transition from a system in which every initial point sinks into one of two wells, to a system where points can pass between the wells. It should be noted that LIME does not pick up on this behaviour. Looking at the feature contribution of x_0 to x_t as a function of γ (Fig. 4.29.),

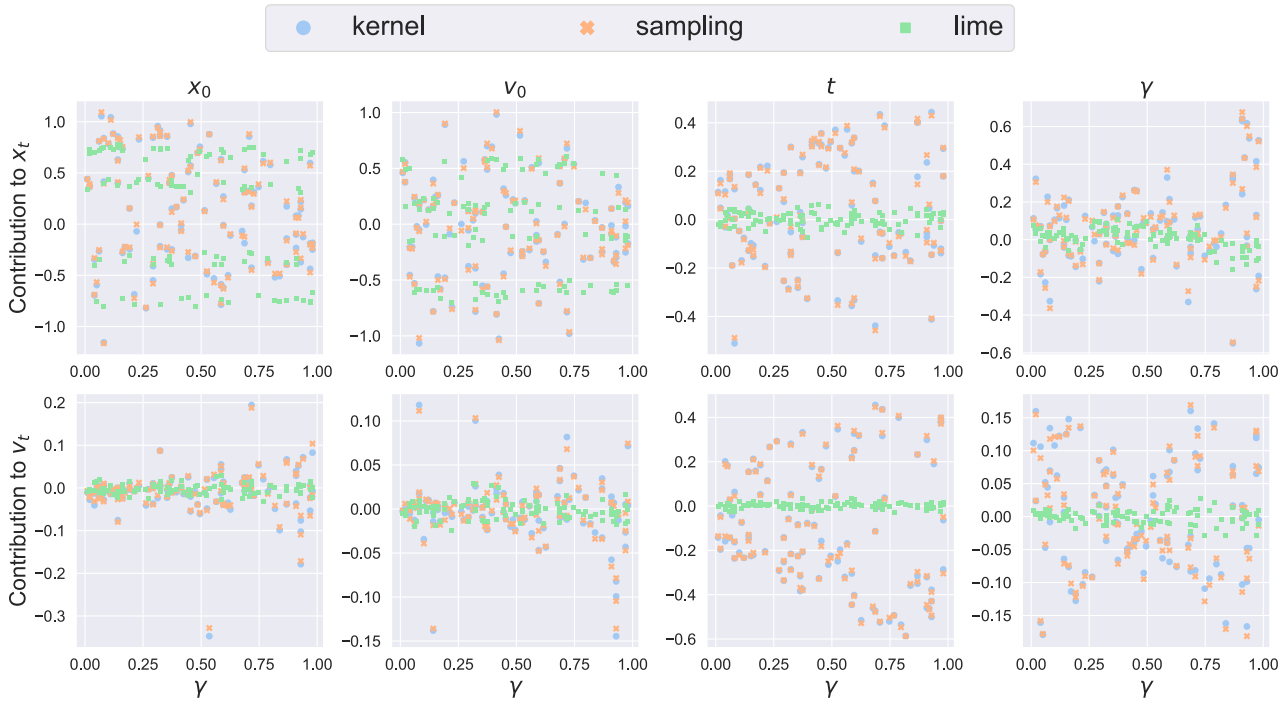


Fig. 4.29: Individual feature attributions for the heavily damped system with variable γ in the simulation. Plotted are the feature attributions (y-axis) of the titular features against γ (x-axis).

shows us that the importance of x_0 and v_0 fall-off with increasing γ as before. The fall off isn't as strong as for the lightly damped system. This is because it is harder to leave these potential wells. Once again, the importance of the time for both labels increases for increasing γ as expected. This reinforces the hypothesis that t is measuring the oscillatory behaviour.

XGBoost

Finally, we can look at the feature importance assigned by XGBoost. This is shown in Fig. 4.30

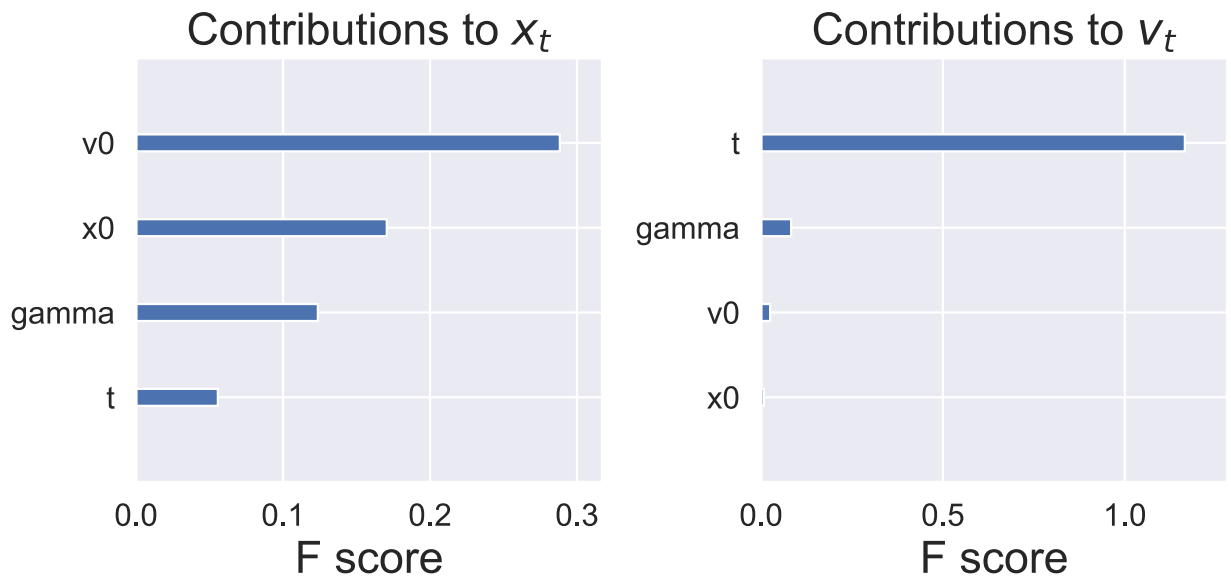


Fig. 4.30: Feature importance (gain) plot of an XGBoost decision tree for the heavily damped double-well with varying γ. This plot used 150 trees with a maximal depth of 10. Left: Feature Attributions to x_t. Right: Feature Attribution to v_t.

This is a good argument for the benefit of individual feature attributions as opposed to just looking at the aggregate importance. Instead of learning about the impact that γ has on the other features, and how it characterises large changes in the system as a whole we get an attribution that simply suggests that it is less important for the predictions than x_0 and v_0. Whilst this aligns with the aggregate feature importance that SHAP suggests we lose the nuance of the individual explainability.

For this case, we do not include a look at the single potential well. The only impact increasing γ has on that system is to increase the amplitude of oscillation as no transit between wells is possible. This means that no interesting changes in the behaviour occur for increased γ.

4.4.3 Section Summary

By looking at the contribution of γ to the labels x_t and v_t we can deduce behaviour like the transition from oscillations around a single well to a trajectory encompassing both wells. This feature could be useful when looking at identifying phase transitions in modelled systems, though these can usually be measured more directly. The feature γ also differed from the other features in that changing it affected the other feature contributions in an intuitive manner. This could be useful in determining model defining parameters. For this purpose, only SHAP was of any use.

Chapter 5

Discussion and Further Work

First, we will discuss the question of whether SHAP and LIME are able to replicate a human's understanding of the system (Sec. 4.1.). Both SHAP and LIME struggle to explain this oscillatory system with light damping with a double-well. They are unable to summarise this system in a way that allows for intuitive human understanding of the system. They replicate a human's understanding in that they do not find an intuitive link between the initial position x_0 and velocity v_0 and (x_t, v_t).

When we apply them to the heavily damped double well, a system for which intuitive explainability exists, both SHAP and LIME are able to attribute the feature importance to the dominating features. LIME is unable to extract nuanced behaviour, such as the oscillations in this system, with the settings used in this work. SHAP can extract this behaviour, though it suffers from aliasing when supplied with insufficient data. SHAP agrees with XGBoost on the aggregate feature importance.

As mentioned in the theoretical section about LIME, LIME has the significant downside that it is so sensitive to hyperparameter tuning. Being sensitive to tuning can be beneficial when dealing with objective truths, such as loss minimisation when training a neural network. But in this case, where we are trying to gain information about the model by applying LIME, it is a detriment. It is a detriment because we can greatly alter the output of the explainer by tuning the kernel width and other parameters. This makes LIME unreliable unless we already have a ground truth explainability. SHAP does not have this problem, it delivers consistent results that align with our expectations. It is able to determine when features do not contribute or contribute very little, to a model's predictions. This statement is true for this fairly simple physical system.

So in answer to the question of which explainability method is more successful we have to respond that SHAP is able to more accurately and reliably replicate a human expert's feature attributions than LIME.

Both SHAP and LIME were able to discern which features contributed to determining which potential well a trajectory ended in. A human looking to extract this information from the data alone would need to look at multiple individual trajectories. Therefore we conclude that both SHAP and LIME were able to extract information from the models that aren't immediately available by looking at the training features and labels.

In setting up this work we asked ourselves whether we could use SHAP and LIME to refine the feature space we need to train and evaluate a model that simulates a physical system. For this scenario with few features, SHAP succeeded at this task by identifying features that bear no importance to the predictions made by the model itself, such as the random feature in Sec. 4.2. LIME was also successful but the success comes with the caveat that relevant features may also be excluded with incorrect parameter tuning. The benefit that explainable machine learning has over human explainability is that it can deal with any number of features. Whilst this is computationally costly, it is as big a problem as it is for a human expert. As such, the next test would be to apply SHAP and LIME to a physical system with a high dimensional feature space. We could then use them to remove the features with the lowest contribution, slimming down the model, with minimal loss in model performance. We can also test the feature refinement on systems with real-world applications.

One application of machine learning in physics is to reduce the experimentation required to gain information about a phenomenon. In the last few decades, researchers trying to uncover high-temperature superconductors started attempting a brute force approach, whereby many materials were tested with minimal preselection as to their likelihood to function as superconductors at anything approaching high temperature [27]. This approach is very costly in terms of manpower as it requires preparation of samples and experimentation for each sample. Using this information to train a neural network to predict likely superconductors [6] is a classification task that has applications in physics. If we can then use explainability to determine which materials to test in order to feed the network more information, then this could be a useful application of explainable AI in physics.

Another question we asked was whether SHAP and LIME can be used as a proxy to judge a model's performance. For this to be possible we require ground truth explainability or some approximation thereof

supplied by a human expert. If the model's feature contributions differ greatly from the ground truth then this suggests that the model is making its predictions for the wrong reasons. In this work, we looked at how the assigned feature contributions differed between a simple SNN and a more complicated DNN. The explainability represented by SHAP and LIME looks only at distributions of data points, but we can see that both the aggregate and the individual feature importance decrease in magnitude for the SNN when it does not capture the behaviour correctly. However, this is because the SNN lays its predictions at the centre of mass. In this case the centre of mass was at $v = 0$, leaving the magnitude of this feature lower than usual.

This question, therefore, requires further testing. For this purpose, we'd need to design a system where a poor model is very likely to suggest values that have high magnitudes and thus high absolute feature importance. This would be a way of examining whether the trend of less accurate models having lower aggregate feature importance across the board is endemic to this model setting or a universal phenomenon.

Another way to test how well we can use SHAP and LIME to judge a model's performance would be to look at how the feature importance changes for predictions the model makes that lie outside the training data set. For example, with our system, we could pass samples for $x_0 = 3$, $v_0 = 3$. The neural networks were only trained on samples that fell within $[-2, 2] \times [-2, 2]$. If the new sample has a vastly different feature importance to the sample at $x_0 = 2$, $v_0 = 2$ then that may be an indication of where the model is going wrong in its predictions of x_t, v_t.

Chapter 6

Conclusion

In this work, we examined how modern explainability frameworks handle the type of regression model that may lend itself to use in a physics setting such as [4]. We applied them to a regression task which aimed to model the Duffing Oscillator. We modelled the Duffing Oscillator with a deep neural network (DNN). The performance of the more complex model was reasonably good for all configurations. A bad model (SNN) was also trained. The predictions made by the DNN were significantly faster than the more conventional time integration (Runge-Kutta) method of predicting a point along the trajectory of the Duffing Oscillator, suggesting that this approach can be used to speed up model simulations for systems where high accuracy isn't as necessary as good prediction speed.

We then applied the explainability systems SHAP and LIME to these models. For a lightly damped system, feature explainability was difficult to judge due to the lack of intuitive human explainability. For a heavily damped system, both SHAP and LIME were able to replicate the intuitive explainability that a human expert would attribute to the system. Applying the feature explanation to individual samples revealed patterns in the data that aren't immediately apparent, such as the impact that the initial position has on the well a trajectory ends up in in the heavily damped case. This relationship is intuitive for this system but this kind of analysis for a more complicated system often requires looking at the data through many different lenses.

From the results, we conclude that SHAP is the more successful framework with which to approach this kind of problem due to it being consistent in its results and its ability to pick out features that don't contribute to the model. LIME was not able to compete in this regard, whilst LIME can be made to suit different problems via its tunability, this is a detriment to the method, as it requires a ground truth which we compare the results to in order to perform said tuning. We also examined XGBoost and concluded that its F-score feature importance aligned with the aggregate feature importance suggested by SHAP. XGBoost does not deliver individual feature importance but can be used to verify the results given by SHAP by looking at the aggregate feature importance. This work reemphasises the issues with LIME touched upon in [23] that suggest that the tunability of LIME may be a weakness and not a strength.

This work shows that explainability in physics has potential in refining the feature space of machine learning models and identifying faulty models, but requires further testing with some stronger applications, such as the explanation of poor predictions in the beamline network [4]. The conclusion we come to is that, whilst these frameworks are unable to completely replace the human expert in interpreting physical systems, they can be used to verify and expand upon a human expert's intuitive explanations of a system. Further, they function as another lens through which to look at data. If used for this purpose they may be helpful in refining neural network designs by identifying extraneous features. As shown in the heavily damped case (Sec. 4.1.) they can be used to give intuitive explanations and these can give a human further understanding of the predictions made by a black-box model.

Bibliography

1. Lundberg, S. M. & Lee, S.-I. in *Advances in Neural Information Processing Systems 30* (eds Guyon, I. *et al.*) 4765–4774 (Curran Associates, Inc., 2017). http://papers.nips.cc/paper/7062-a-unified-approach-to-interpreting-model-predictions.pdf.
2. Ribeiro, M. T., Singh, S. & Guestrin, C. "Why Should I Trust You?": Explaining the Predictions of Any Classifier. *CoRR* **abs/1602.04938.** arXiv: 1602.04938 (2016).
3. Duffing, G. *Erzwungene Schwingungen bei veränderlicher Eigenfrequenz und ihre technische Bedeutung.* (Braunschweig, F. Vieweg & Sohn, 1918).
4. Bellotti, R. *Accelerating Accelerators: Fast Surrogate Models for Beam Prediction* MA thesis (ETH Zurich, 2020).
5. Chen, T. & Guestrin, C. XGBoost: A Scalable Tree Boosting System. *CoRR* **abs/1603.02754.** arXiv: 1603.02754 (2016).
6. Konno, T. *et al.* Deep Learning Model for Finding New Superconductors. *Phys. Rev. B* **103,** 014509 (1 Jan. 2021).
7. Loyola-González, O. Black-Box vs. White-Box: Understanding Their Advantages and Weaknesses From a Practical Point of View. *IEEE Access* **7,** 154096–154113 (2019).
8. Lundberg, S. M. *et al.* Explainable machine-learning predictions for the prevention of hypoxaemia during surgery. *Nature Biomedical Engineering* **2,** 749 (2018).
9. Ben-Gal, I. in *Encyclopedia of Statistics in Quality and Reliability* (American Cancer Society, 2008). ISBN: 9780470061572. eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1002/9780470061572.eqr089.
10. Chollet, F. Xception: Deep Learning with Depthwise Separable Convolutions. *CoRR* **abs/1610.02357.** arXiv: 1610.02357 (2016).
11. Molnar, C. *Interpretable Machine Learning: A Guide for Making Black Box Models Explainable.* (https://christophm.github.io/interpretable-ml-book/, 2021).
12. Ribeiro, M. T. *LIME Documentation* https://github.com/marcotcr/lime. Last Accessed: 10-10-2021, Version: 0.2.0.1. July 2021. https://github.com/marcotcr/lime.
13. Shapley, L. S. in *Contributions to the Theory of Games (AM-28), Volume II* (eds Kuhn, H. W. & Tucker, A. W.) 307–318 (Princeton University Press, 2016). https://doi.org/10.1515/9781400881970-018.
14. Lundberg, S. *SHAP Documentation* https://shap.readthedocs.io/en/latest/index.html#welcome-to-the-shap-documentation. Last Accessed: 10-10-2021, Version: 0.39.0. 2018. https://shap.readthedocs.io/en/latest/index.html%5C#welcome-to-the-shap-documentation.
15. Strumbelj, E. & Kononenko, I. An Efficient Explanation of Individual Classifications using Game Theory. *Journal of Machine Learning Research 11 (2010) 1-18* (2009).
16. Dobrushkin, V. *MATHEMATICA Tutorial, Part 2.3: Duffing Oscillator* https://www.cfm.brown.edu/people/dobrush/am34/Mathematica/ch3/duffing.html. Last Accessed: 09-10-2021.
17. *Wikipedia: Duffing Equation* Last Accessed: 10-10-2021. %5Curl%7Bhttps://en.wikipedia.org/wiki/Duffing_equation%7D (2021).
18. Virtanen, P. *et al.* SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python. *Nature Methods* **17,** 261–272 (2020).
19. Dormand, J. & Prince, P. A Family of Embedded Runge-Kutta Formulae. *Journal of Computational and Applied Mathematics* **6,** 19–26. ISSN: 0377-0427. https://www.sciencedirect.com/science/article/pii/0771050X80900133 (1980).
20. Martín Abadi *et al. TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems* Software available from tensorflow.org. 2015. https://www.tensorflow.org/.

21. Pedregosa, F. *et al.* Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research* **12,** 2825–2830 (2011).

22. Batchelor, A. *NN Lorenz Lobes rev A* `https://github.com/aibatchelor22/Applied-Math/tree/master/Neural-Networks`. Last Accessed: 08-10-2021. 2019.

23. Altmann, T. *et al. Limitations of Interpretable Machine Learning Methods* (eds Molnar, C. *et al.*) (bookdown, 2019).

24. Altmann, T. *et al. Limitations of Interpretable Machine Learning Methods* (eds Molnar, C. *et al.*) (bookdown, 2019).

25. Kingma, D. P. & Ba, J. Adam: A Method for Stochastic Optimization. *CoRR* **abs/1412.6980** (2015).

26. Waskom, M. L. seaborn: statistical data visualization. *Journal of Open Source Software* **6,** 3021. `https://doi.org/10.21105/joss.03021` (2021).

27. Jin, K. *et al.* Combinatorial search of superconductivity in Fe-B composition spreads. *APL Materials* **1,** 042101. eprint: `https://doi.org/10.1063/1.4822435` (2013).

Appendix A

Appendix A

In this appendix we include the network training plots for the different systems that weren't detailed in the main body of the work.

Random: Heavily Damped Double-Well

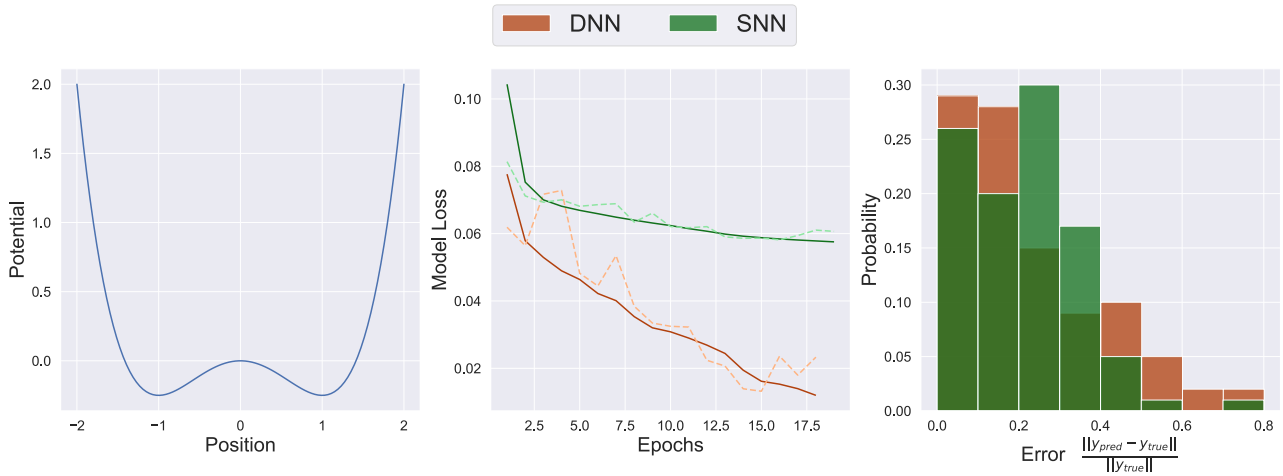


Fig. A.1: Here are shown the model training and validation accuracy for the neural networks in the configuration ($\alpha = -1$, $\beta = 1$, $\gamma = 0.37$, $\delta = 1.0$, $\omega = 1.2$) potential when we use a random training feature. The potential is shown on the left, in the centre are the model training and validation loss for the simple and complex models. On the right is an error plot that compares the error on the validation set for the two models, outliers, the largest 5% of the error, are removed.

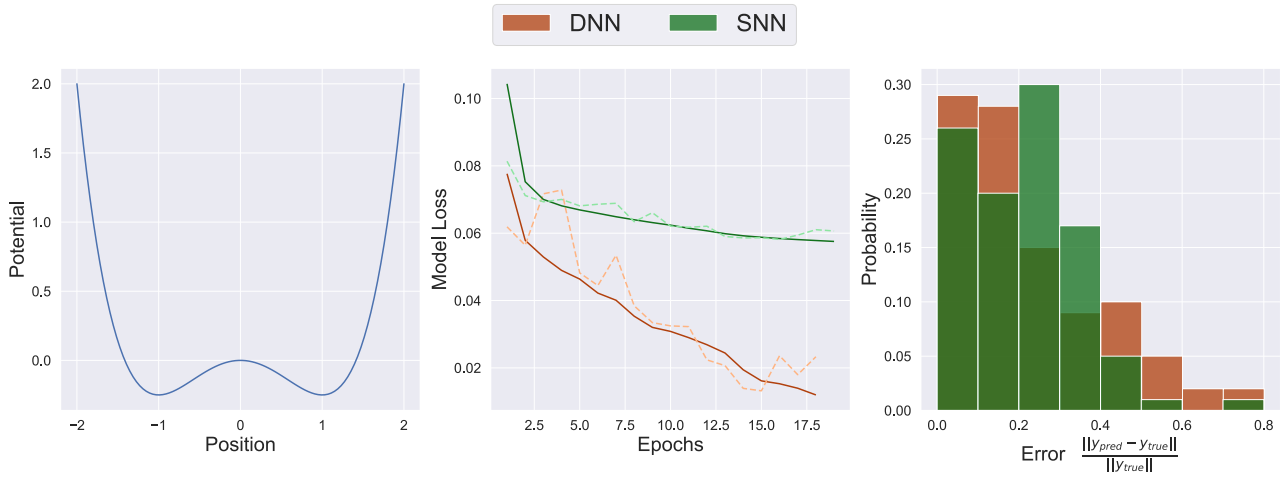


Fig. A.2: Predictions made by the machine learning models for the damped double well potential ($\alpha = -1, \beta = 1.0, \gamma = 0.37, \delta = 1.0, \omega = 1.2$) with a random training feature; Left: Plot of the model outputs; Centre: Plot of $x(t)$ by t. Right: Plot of $v(t)$ by t.

Energy: Heavily Damped Double-Well

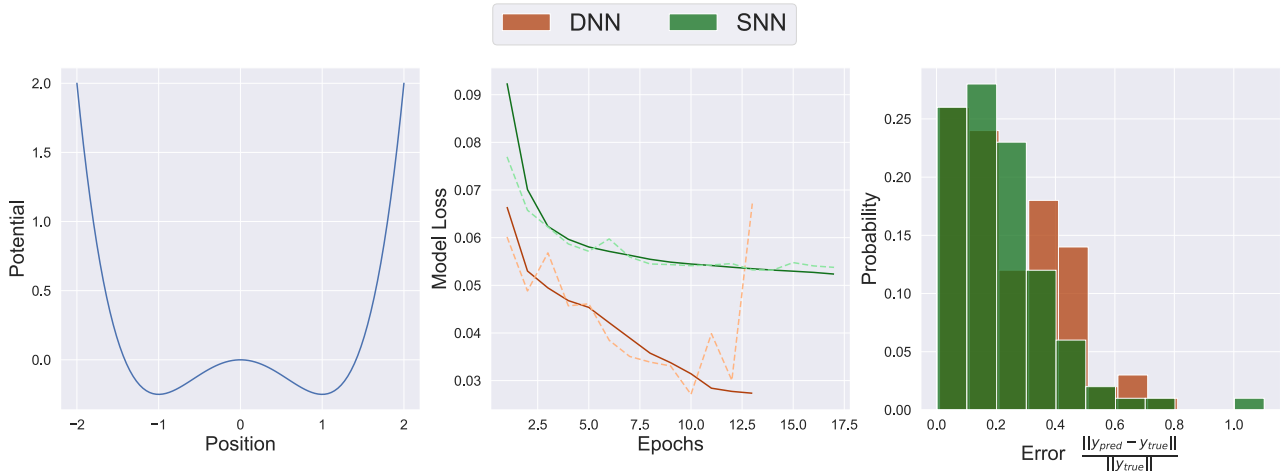


Fig. A.3: Here are shown the model training and validation accuracy for the neural networks in the configuration ($\alpha = -1$, $\beta = 1$, $\gamma = 0.37$, $\delta = 1.0$, $\omega = 1.2$) when we the initial system energy as a training feature. The potential is shown on the left, in the centre are the model training and validation loss for the simple and complex models. On the right is an error plot that compares the error on the validation set for the two models, outliers, the largest 5% of the error, are removed.

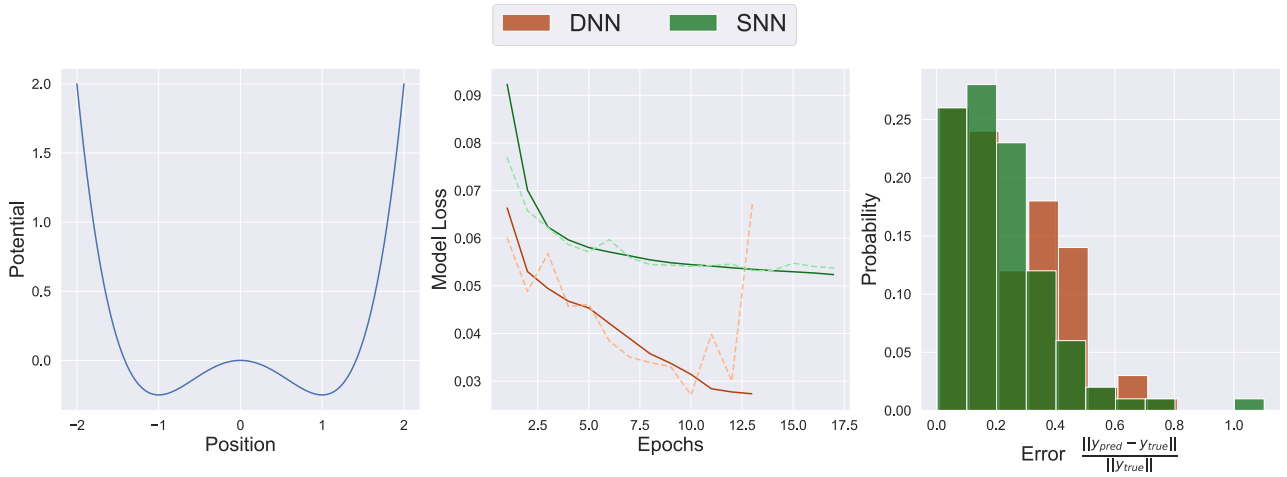


Fig. A.4: Predictions made by the machine learning models for the damped double well potential ($\alpha = -1, \beta = 1.0, \gamma = 0.37, \delta = 1.0, \omega = 1.2$) with the initial system energy as a training feature; Left: Plot of the model outputs; Centre: Plot of $x(t)$ by t. Right: Plot of $v(t)$ by t.

Gamma: Heavily Damped Double-Well

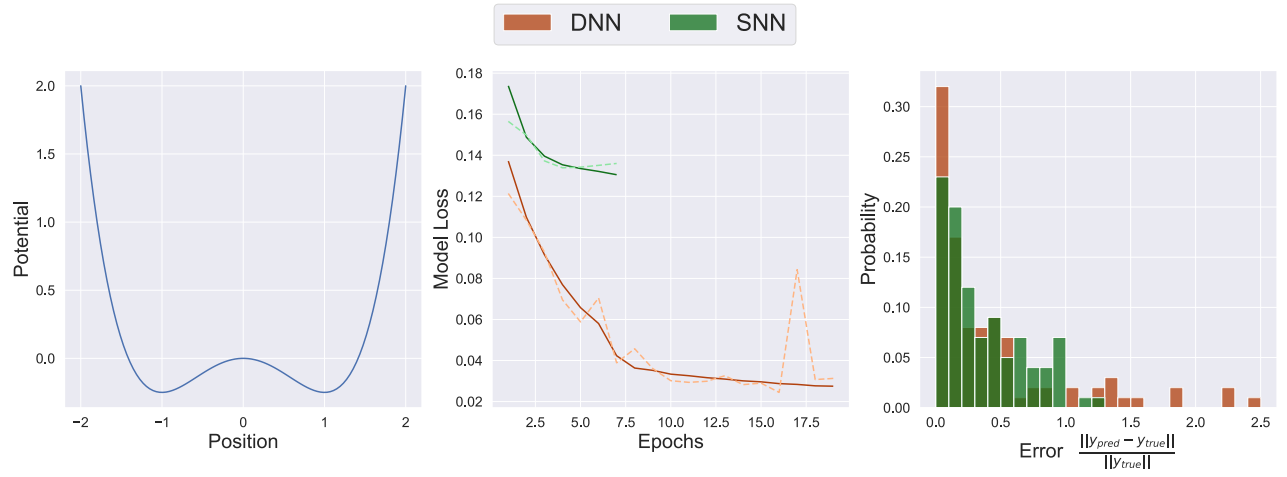


Fig. A.5: Here are shown the model training and validation accuracy for the neural networks in the configuration ($\alpha = -1$, $\beta = 1$, $\gamma = 0.37$, $\delta = 1.0$, $\omega = 1.2$) when we use the amplitude of the driving force (γ) as a training feature. The potential is shown on the left, in the centre are the model training and validation loss for the simple and complex models. On the right is an error plot that compares the error on the validation set for the two models, outliers, the largest 5% of the error, are removed.

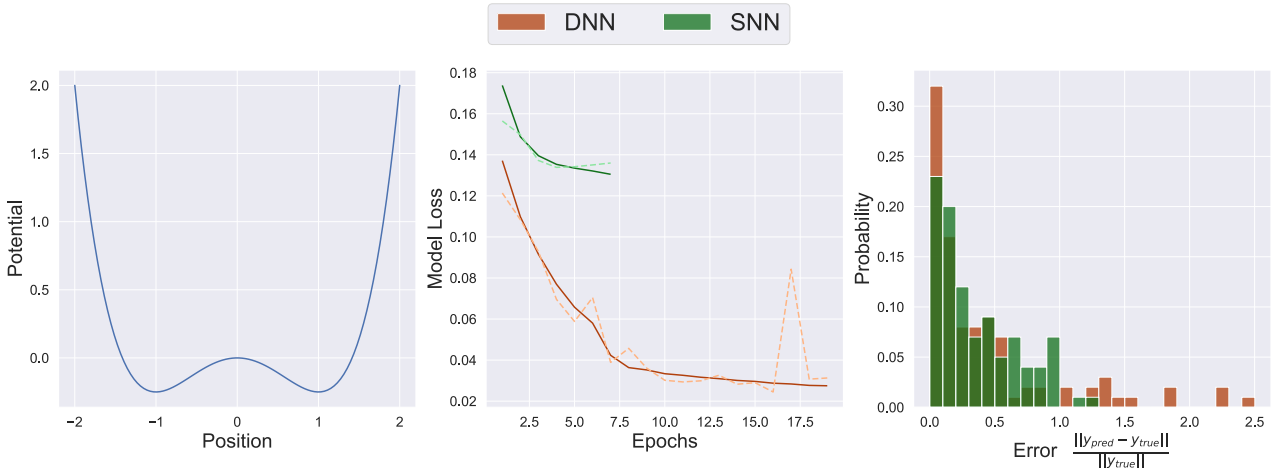


Fig. A.6: Predictions made by the machine learning models for the damped double well potential ($\alpha = -1, \beta = 1.0, \gamma = 0.37, \delta = 1.0, \omega = 1.2$) where we use the amplitude of the driving force (γ) as a training feature; Left: Plot of the model outputs; Centre: Plot of $x(t)$ by t. Right: Plot of $v(t)$ by t.

Gamma: Single Well

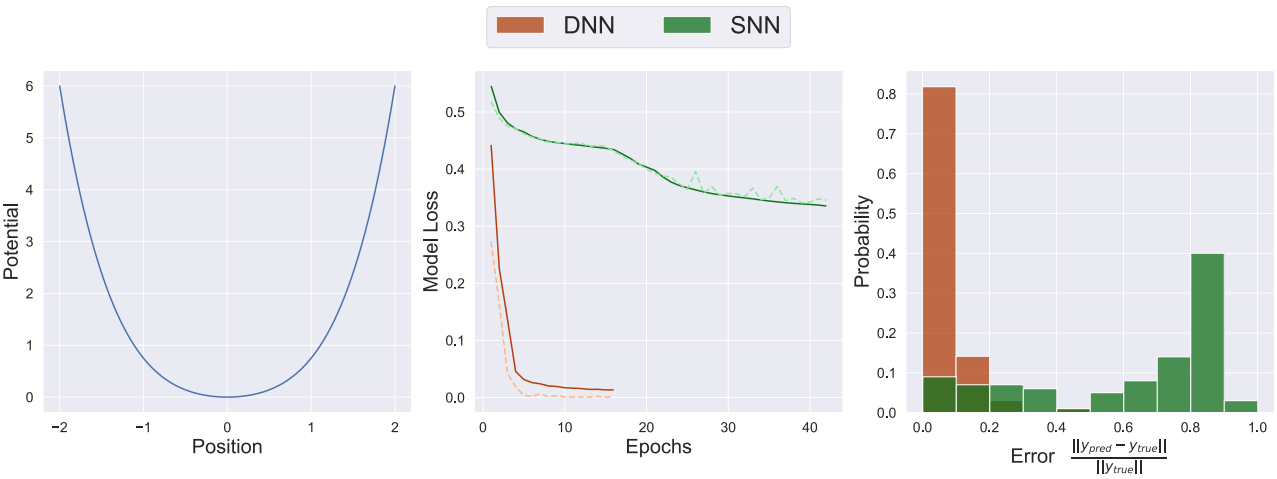


Fig. A.7: Here are shown the model training and validation accuracy for the neural networks in the configuration ($\alpha = 1$, $\beta = 1$, $\gamma = 0.37$, $\delta = 1.0$, $\omega = 1.2$), when we use the amplitude of the driving force (γ) as a training feature. The potential is shown on the left, in the centre are the model training and validation loss for the simple and complex models. On the right is an error plot that compares the error on the validation set for the two models, outliers, the largest 5% of the error, are removed.

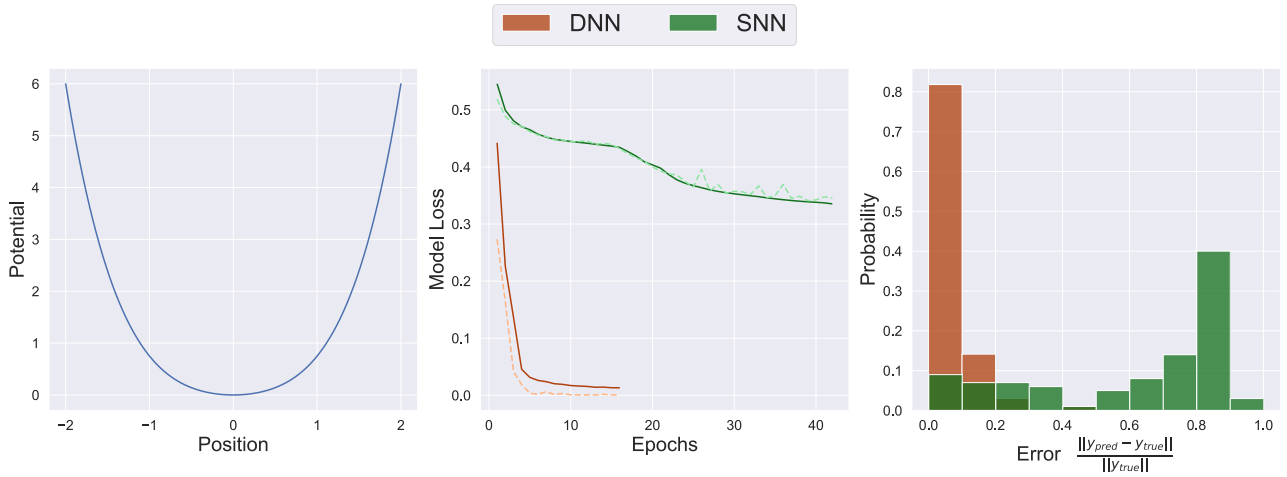


Fig. A.8: Predictions made by the machine learning models for the single well potential ($\alpha = 1, \beta = 1.0, \gamma = 0.37, \delta = 1.0, \omega = 1.2$) where we use the amplitude of the driving force (γ) as a training feature; Left: Plot of the model outputs; Centre: Plot of $x(t)$ by t. Right: Plot of $v(t)$ by t.

Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Declaration of originality

The signed declaration of originality is a component of every semester paper, Bachelor's thesis, Master's thesis and any other degree paper undertaken during the course of studies, including the respective electronic versions.

Lecturers may also require a declaration of originality for other written papers compiled for their courses.

I hereby confirm that I am the sole author of the written work here enclosed and that I have compiled it in my own words. Parts excepted are corrections of form and content by the supervisor.

Title of work (in block letters):

Model-Agnostic Machine Learning Explainability for Regression Tasks in Physical Systems

Authored by (in block letters):

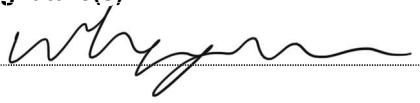
For papers written by groups the names of all authors are required.

Name(s):	**First name(s):**
Grosche	Friedrich Wilke

With my signature I confirm that

- I have committed none of the forms of plagiarism described in the 'Citation etiquette' information sheet.
- I have documented all methods, data and processes truthfully.
- I have not manipulated any data.
- I have mentioned all persons who were significant facilitators of the work.

I am aware that the work may be screened electronically for plagiarism.

Place, date	**Signature(s)**
Zurich, 11/10/2021	

For papers written by groups the names of all authors are required. Their signatures collectively guarantee the entire content of the written paper.